Towards a theory for the emergence of grid and place cell codes

by

Tzuhsuan Ma

Bachelor of Science in Physics

Submitted to the Department of Brain and Cognitive Sciences
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2020

Signature redacted

Author .....
Department of Brain and Cognitive Sciences
November 26, 2019

Signature redacted

Certified by .....
Matthew A. Wilson
Sherman Fairchild Professor of Neuroscience and Picower Scholar
Thesis Supervisor

Signature redacted

Accepted by .....
Rebecca Saxe
John W Jarve (1978) Professor of Brain and Cognitive Sciences
Associate Head, Department of Brain and Cognitive Sciences
Affiliate, McGovern Institute for Brain Science

Towards a theory for the emergence of grid and place cell codes

by

Tzuhsuan Ma

Submitted to the Department of Brain and Cognitive Sciences
on November 26, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This work utilizes theoretical approaches to answer the question: *which functions grid and place cells perform that directly lead to their own emergence?* To answer such a question, an approach that goes beyond a simple modelling is necessary given the fact that there could be circuit solutions other than grid or place cells that better perform these functions. With this reasoning, I adopted a systematic guideline that aims for an *optimization principle* attempting to find the optimal solution for performing the hypothesized functions while reproducing the correct phenomenology.

Within the optimization principle framework, I applied both recurrent neural network (RNN) training and coding-theoretic approaches to set up appropriate optimization problems for testing a given function hypotheses. The descriptive function hypotheses: 1) *Grid cells exist for having a high-capacity and robust path-integrating code* and 2) *Place cells exist for having a sequentially-learnable and highly-separable path-integrating code* were adopted. The non-converging performance in training an RNN to perform a hard navigation task suggests that the attractor dynamics forbids a network to simultaneously possess online learnability and high coding capacity. Because of this dynamical constraint in learning, a grid cell circuit has to be hardwired through some developmental process and cannot be easily modified by an experience-based synaptic rule without compromising its capacity. On the contrary, a place cell circuit being able to continually learn a novel environment inevitably have a mere linear capacity. These results imply that the functional separation of grid and place cell systems observed in the brain could be a result of an unavoidable dynamical constraint from their underlying RNNs.

Lastly, a fundamental principle called the *tuning-learnability correspondence* was uncovered in pursuit of a sequentially learnable neural implementation for place cells. It explains that the seemingly incidental existence of conjunctive tuning property is in fact caused by a necessary metastable attractor dynamics for having sequential learnability rather than by another functional need attached to a particular tuning property. In addition, from the unique property of metastable attractor dynamics, I also predicted that the biased place field propensity recently observed in CA1 sub-

region should originate from CA3 due to an inevitable biased activation in the RNN as a side effect of such a dynamical property. In sum, both this principle and the subsequent prediction thus provide a new perspective that contradicts the conventional wisdom which often assumed that a certain nonspatial tuning property exists for performing a relevant task.

Thesis Supervisor: Matthew A. Wilson
Title: Sherman Fairchild Professor of Neuroscience and Picower Scholar

Acknowledgments

I thank my advisor, Ila Fiete, for her valid challenges and encouragements that helped me both venture and think critically when I was new in the lab and to neuroscience; I also thank her for all the discussions regarding my research, this thesis will not be possible without her contribution. I thank her husband, Gregory Fiete, for pointing me a direction towards neuroscience when I was facing such an uncertainty in career change. It was for them, I can ever be a theorist in the field of neuroscience.

I thank my former advisor, Gennady Shvets, for his advises when I was in physics.

I thank my committee, Matt Wilson, Rebecca Saxe, Laura Colgin, and Sasha Rakhlin for their continuous supports for my graduation. I thank Matt and Laura for instructing and guiding me on writing the thesis. Because of their guidance and kindness, this journey became one of the most enjoyable time in my PhD.

I thank people who were in the Fiete lab: Birgit, Ingmar, Rishi, John, Berk, Biraj, Ryota, Abhranil, Anastasiya, Lixiang, Chris, and Kalina for our many enjoyable discussions in science and life. I thank Rishi and John for their warm supports, encouragement, and being kind when I started to doubt.

I thank my friends in Boston, Manyi, Su, Mikail, Eunice, Mirko, Leenoy, David, and C.J. for being here with me in the final stage of my PhD. I also thank Erik, Evan, and Jason for listening to me and helping me tremendously.

I thank my friends in Austin, Haisoyu, Lucy, Wanhsuan, Sariel, Yu-An, Yikuan, Weijin, Benny, Xing, Xi, Danlu, Panpan, Zhiyuan for their endless caring and the laughter we shared. I thank Ernie and Kalina for being with me constantly even after I left Austin. And I thank Luda for all the light brought into my life.

I thank my mom, my dad, and my brother for witnessing my stories and for always finding some goodness in it. I thank my partner Weimien for what has been, and will be.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The discovery of place cells and grid cells marked an important step in studying neural basis of cognition. Both place and grid cells encode *position* with precision; they are only active when the animal is at either one or multiple particular locations. Most amazingly, these cells even do so when the surrounding sensory cues are seriously deprived. In contrast to the earlier discovered cell types with their activities mostly representing sensory stimuli, the circuitry that underlies place or grid cells must perform computation that involves *memory* and, subsequently, *integration* and *inference*—in other words, the essential computational ingredients for cognition. For this reason, place and grid cells could be important model systems for studying cognition down to the circuit level.

Curiously enough, although both place and grid cells are spatial cells, their tuning curves and dynamics are distinctly different. It raises the question—why isn't there just one type of spatial cells if position coding is the only goal? To bring some insights into this question, I hypothesized the major functions of place and grid cells based on the previous studies, and tested these hypotheses using a theoretical framework called *optimization principle*. Thus, the main objective of this thesis is to find out precisely what functions place and grid cells perform that necessarily lead to their own emergence. We will see that, in fact, place and grid cells provide two spatial codes that are *complementary* to each other and both are essential for the brain to perform spatial tasks.

In this chapter, I will start with a brief review on the tuning properties and circuitries of grid and place cells from both experimental and theoretical perspectives. Before going into any specific function hypotheses, to be discussed in Chapter

2 and 3, I will state a fundamental stance of this thesis—which assumes that grid and place cells are functionally complementary—along with its implication on the circuitry. I will then introduce the framework of optimization principle and point out how it differs from a more conventional methodology of hypothesis testing: *theoretical modeling*. Lastly, I will show that the framework of optimization principle well complements Marr's three levels of analysis with an additional emphasis on neural implementations and learning dynamics.

1.1 Phenomenology

Place and grid cells display both slow and fast dynamics. The slow dynamics, or firing rate dynamics, on the time scale of a second and above, determines their tuning properties. On the other hand, the fast dynamics, or spike dynamics, on the time scale of small fractions of a second and below, produces effects like theta or gamma oscillations, phase precessions, sharp-wave ripples, or sequences (Colgin, 2016). The focus of this thesis and the function hypotheses to be discussed are based on phenomena from the slow dynamics. Therefore, the following introduction will be limited mainly to the tuning properties and to the circuit mechanism based on the canonical neural network rate equation (Ermentrout and Terman, 2010).

1.1.1 Tuning properties of grid and place cells

Although the firing rate of a cell, or its activity, is changing rapidly over the course of seconds, the tuning curves that underlies it are stable over the course of minutes, hours, or even days. For our purposes in later formulating function hypotheses, it is useful to categorize a tuning property to be either *static* or *dynamic*. A static tuning property concerns a relatively fixed tuning shape of a cell within the course of a few minutes, whereas a dynamic tuning property focuses on the change of each tuning curve as well as the change of cofiring structure at the population level over the course of tens of minutes to hours.

Static tuning properties of grid cells

Periodic tuning curves with locally narrow fields. As first discovered by Hafting et al. (2005), a grid cell in medial entorhinal cortex (MEC) of a freely moving rat fired a

burst of spikes only at certain locations within a small square box; the multiple firing locations eventually resembled a triangular lattice after the rat explored the entire box as shown in Figure 1.1(b); the identical local fields at lattice points is a narrow gaussian function with a width 30cm. The periodic tuning curve of a grid cell can be well described by von Mises functions (Mathis et al., 2013).

Multi-periodicity and Modularity. The scale of the triangular lattice of a grid cell ranges from 25cm to 3m (Brun et al., 2008) (but the range is more likely to be from 30cm to 1m (Stensola et al., 2012)). The spacing of a grid cell progressively increases from dorsal to ventral MEC as shown in Figure 1.1(b,c). Surprisingly, however, Stensola et al. (2012) found that this increasing of grid spacing is not continuous for 62 simultaneously recorded grid cells; instead, only three or four particular periods exist for a single rat. In other words, an ensemble of grid cells in MEC forms discrete modules with different periods.

Constant grid-spacing ratio. Moreover, the ratios of the period of two adjacent modules are approximately a constant, $p_i/p_{i+1} \approx 1.4$, across different rats (Stensola et al., 2012).

Uniform phase coverage within a module. Because the grid cell tuning curves are periodic, one can assign a unique phase combination for every tuning curves within a module (two phases for a 2D periodic function). Given that one can assign an arbitrarily chosen tuning curve to be zero-phase, other tuning curves with non-zero phase correspond to shifted versions—as shown in Figure 1.1(d). It turns out that the grid cells within a module have uniform distribution in their phases such that every part of space in the box is equally covered (Hafting et al., 2005).

Dynamic tuning properties of grid cells

Realignment and conserved cofiring structure during sleep. The dynamic tuning property of a grid cell ensemble was revealed when a rat was moved to a different environment. As shown in Figure 1.1(e), the tuning curves underwent both a translation and rotation (Fyhn et al., 2007) after the movement from a square to a circular environment—a phenomenon termed the grid cell realignment. Crucially, the cells that cofired within a module realigned in the same way such that their tuning correlation was preserved. Amazingly, even during sleep, this cofire structure among all cells within a module was preserved as well (Trettel et al., 2019; Gardner et al., 2019). Because that the grid cell realignment can be easily explained away by a simple reset

on the default orientation and displacement of tuning curves when an animal enters a new environment, these experimental findings strongly suggest a static nature of grid cell tuning curves. In other words, both the tuning curves and the circuitry underlies them remain largely unchanged over experiences.

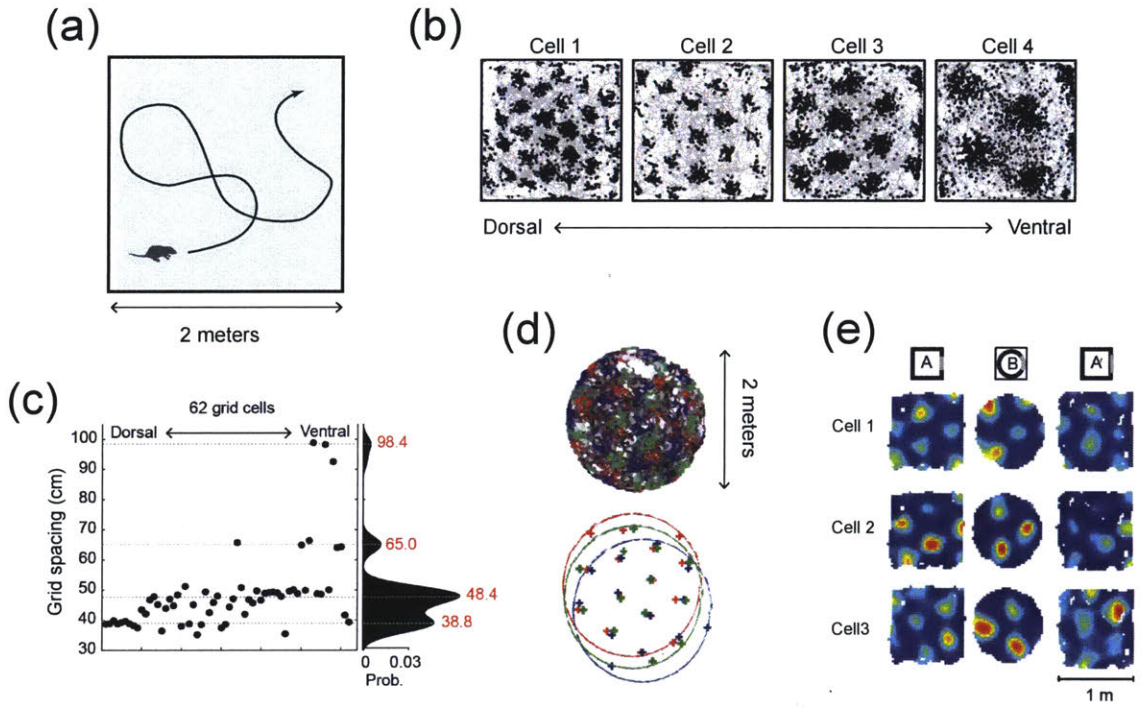

Figure 1.1: Grid cell tuning properties. **(a)** Multiple cells in a rat's medial entorhinal cortex (MEC) are simultaneous recorded during free exploration in a small environment. **(b)** Single grid cell tuning curves exhibit a triangular lattice arrangement. The spacing between any two neighboring local fields is smaller in dorsal MEC and increases towards ventral MEC. **(c)** Grid spacing statistics from 62 simultaneous recorded cells shows a multimodal distribution. It revealed that a population of grid cells forms multiple modules with spacing ranging roughly from 40 cm to 1m. **(d)** Within a module, all grid cell tuning curves differ only by a 2D translation which can be specified by two phases. The phases are uniformly distributed from 0 to '2pi such that all grid fields together tile space uniformly. **(e)** Grid cell realignment happened after moving the rat from a square room to a circular room. it rotated and translated a cell's original tuning curve without changing the tuning shape. The cells belong to the same module realign together such that their cofiring structure is preserved. (b,c) are replotted from (Stensola et al., 2012), (d) from (Hafting et al., 2005), and (e) from (Fyhn et al., 2007).

Static tuning properties of place cells

Narrow unimodal tuning curves in a small environment. In the hippocampus of a rat, a place cell have a spatially selective firing field of a gaussian shape with a narrow

width 30cm located somewhere in a relatively small environment (O'Keefe and Nadel, 1978) as shown in Figure 1.2(b).

Uniform spatial coverage. A place cell ensemble in one subregion (e.g. CA1 or CA3) has their fields tiling entire environment with roughly a uniform coverage (Muller et al., 1987). This tuning property together with the narrow unimodal tuning shape show that there should be only a small fraction of cells that fire at any time because each cell only fires in a small region within the environment. For example, it has been recorded that there are 3% of active cells (with a rate threshold 1.5 Hz) at any time in CA3 subregion (Leutgeb et al., 2004).

Different tuning widths. Similar to grid cells, place cells also have multiple scales in their tuning width that progressively increases from dorsal to middle with an average ranging from 24cm to 42cm (Maurer et al., 2005)—as shown in Figure 1.2(c) with two example cells. However, unlike grid cells, there is no sign of modularity in a place cell ensemble.

Multimodal tuning curves in a large environment with biased field propensity. In a large environment, a place cell eventually fires at multiple locations. Rich, Liaw and Lee (2014) demonstrated, with 253 simultaneously recorded cells, that most cells display many fields in a 45m long track as shown in Figure 1.2(d). The multimodal tuning property immediately implies that the resource of a place cell ensemble is limited, in which the unused cells eventually run out so that the cells need to be reused for encoding new locations. In such a reasoning, every cell should be activated equally likely in order to maximize coding capacity. Surprisingly, the number of place fields for each cell is far from a constant. Rather, they follow a Gamma-Poisson distribution for which more than one-third of the cells are completely silent on the one extreme, and a few cells have more than twenty fields on the other extreme (Rich et al., 2014).

Dynamic tuning properties of place cells

Rapid formation of place fields in a novel environment. When a rat first introduced to a novel environment, the firing field of a place cell is unstable. But after exploring in a relatively small environment for 10 mins (about revisiting every location 3 or 4 times), a set of the place fields, or a map, are stabilized (Wilson and McNaughton, 1993).

Retaining previous learned place fields in a familiar environment. When a rat returns

to a familiar environment, the same set of place fields will immediately be reactivated indicating that place cell ensemble has a way to store such a map in its connectivity. More importantly, a place cell ensemble can continuously learning and remembering multiple environments (Alme et al., 2014).

Remapping. If one compares a newly learnt map to an old one, an individual cell fires at a different location in different environments. More importantly, the cofiring structure among all place cells also changed to such an extreme that the firing location of each cell randomized, a phenomenon named global remapping (Muller et al., 1987). It's been shown that the hippocampus can store up to eleven maps with little correlations between any of the two maps (Alme et al., 2014). The hippocampal remapping also comes with the other classes (see a comprehensive review (Leutgeb et al., 2007) for more details); another prominent class is called rate remapping for which the peak firing rate of individual cells change instead of the firing location.

Reactivation of recent firing patterns during sleep. During sleep, place cells are still active with their firing pattern resembles that during awake to a certain extent. More specifically the cofiring structure of a place cell ensemble resembles that of the recent experience but not prior to that (Wilson and McNaughton, 1994) and this structure decays rapidly during subsequent sleep (Kudrimoti et al., 1999).

1.1.2 Circuitry of the dual system

Studying circuitry in the brain is helpful for gaining insights about the mechanism that might underlie a certain neural activity. It is especially helpful for distinguishing which aspects of a given phenomenon are responsible for a functional need or simply exist because of biological constraints. A few experimental and theoretical investigations have revealed certain aspects regarding what circuitry underlies grid or place cells separately. More interestingly, the two circuits have also been studied together to better understand the functions of such a dual system. But so far, the reason for the brain to have both systems remains unknown. Below, I will briefly talk about the circuits of grid and place cells separately, and then I will discuss two circuit paradigms for the dual systems—i.e., either place-centric or grid-centric paradigm. I will then point out the issues of such paradigms for the fact that 1) they violate the function complementarity standpoint of this thesis and 2) for some important contradictions to experimental findings.

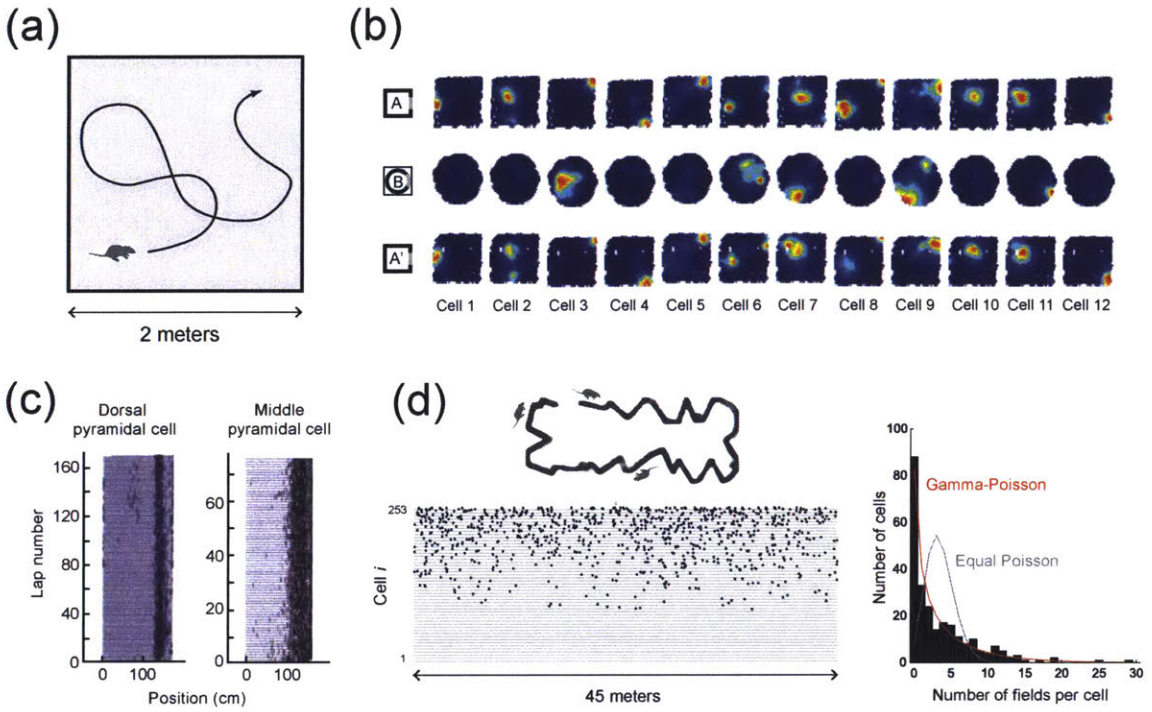

Figure 1.2: Place cell tuning properties. **(a)** Multiple cells in a rat's hippocampus are simultaneous recorded during free exploration in a small environment. **(b)** Single place cell tuning curves exhibit a unimodal narrow field. The fields from the population of cells tiles entire room uniformly. When the rat was introduced to a new environment (from square to circular room), a new set of fields emerge after 10-min exploration, and a global remapping happened where both the firing locations and the cofiring structure among all cells are not preserved. When the rat is reintroduced to a familiar environment (square room), the place fields retained. **(c)** Place field size progressively increases from dorsal to middle with an average ranging from 24cm to 42cm. **(d)** In a large environments, multiple place fields with a biased field propensity was observed. The distribution of the number of fields for 253 cells is Gamma-Poisson instead of regular Poisson in the case if all cells have the same number of fields on average. (b) is replotted from (Fyhn et al., 2007), (c) from (Maurer et al., 2005), and (d) from (Rich et al., 2014).

Grid cell circuit: an RNN with topographic uniform local connections

Theoretically, it has been shown that an RNN—with uniform local connections (i.e., a cell connects to its neighbors in exactly the same way as everyone else does) and a nonlinear activation (capable of creating versatile attractor geometries)—possesses a continuous attractor with each state on the attractor that resembles a periodic activity pattern (Fuhs, 2006; Burak and Fiete, 2009). If a feedforward velocity input is properly incorporated, the continuous attractor can path-integrate such that a periodic activity pattern (of multiple active bumps) moves within the neural ensemble precisely tracking the animal's position in real time. Consequently, a cell will start to

fire whenever an activity bump moves towards it. And since the bumps are arranged periodically, the cell will have a periodic tuning curve just like a grid cell.

This uniform local connectivity underlies, so far, all the slow dynamics of a grid cell ensemble including a preserved cofiring structure within a module across environments even after grid realignments (Yoon et al., 2013), as well as the preserved cofiring structure during sleep (Trettel et al., 2019; Gardner et al., 2019). Even in a non-circular or a non-square environment when grid cells display a highly distorted tuning curves, the cofiring structure is still preserved (Barry et al., 2007; Stensola et al., 2015). Most recently, a more direct evidence also showed a topographic organization of grid cells on a two-dimensional neural sheet (Gu et al., 2018). For which the neighbor cells belong to the same module, have similar firing pattern, and the dropping and rising of similarity, while moving on the neural sheet, follows closely with the tuning curves from a continuous attractor network with uniform local connections.

Place cell circuit: an RNN with non-topographic connections

Unlike grid cells, there is no topographic organization in the recurrent connectivity of CA3 place cells (O'Keefe et al., 1998; Redish et al., 2001; Guzman et al., 2016) with, on average, one out of 300,000 CA3 cells receiving inputs from the other 12,000 CA3 cells (Witter and Amaral, 2004; Rolls, 2013). The apparent random connectivity is closely related to the global remapping property due to the fact that a single fixed set of connections (after learning multiple environments) can generate multiple uncorrelated sets of place fields; that is, two cells with similar tuning curves in one environment could have very different tuning curves in another one such that it is impossible to have a topographic organization among CA3 cells.

In theory, multiple sets of place fields can also be generated from an attractor network called the *multichart attractor network* (McNaughton et al., 1996; Samsonovich and McNaughton, 1997; Tsodyks, 2005) which stores multiple uncorrelated continuous attractors. The connectivity of a multichart attractor network is random (and symmetric) which agrees with the experimental findings. However there are many unresolved issues regarding the biological implausibility from such a circuitry and from the dynamics it generates as I will address and discuss in detail in Chapter 3 (Section 3.4).

Grid- or place-centric paradigm

Having known the connectivity of individual circuits, I will now discuss two circuit paradigms for the dual system. In a grid-centric paradigm, the grid cell representation is assumed to be functionally relevant to behaviors directly, whereas the place cell representation is some intermediate step for finally transforming sensory or self-motion cues to the grid cell representation. The corresponding circuit thus consists of a place-to-grid feedforward projection as shown in Figure 1.3(a). Similarly, in a place-centric paradigm, the circuit consists of a grid-to-place projection as shown in Figure 1.3(b). The focus of the past theoretical works have been only to understand how this feedforward projection gives rise to the downstream target representation given that the upstream representation already exists. One issue of these studies is that they did not provide any insights to the questions: 1) how does the upstream circuit come about, and 2) what exactly are the functions needed to perform such that the downstream representation is required. The purpose of this thesis is to gain insights regarding these questions by searching for the emergence based on the hypothesized functions (to be discussed in Section 1.2). Below, I pointed out the inconsistencies between these two circuit paradigms and the experimental results.

Place-to-grid architecture

The motivation behind this architecture was based on some experimental findings that 1) place cells emerge few weeks prior to grid cells during their development (Langston et al., 2010; Wills et al., 2010) and 2) grid cells lose their triangular firing pattern after hippocampal inactivation (Bonnevie et al., 2013). However, none of those observations provides a functional reason for such an architecture. To motivate it on the basis of functions, Dordek et al. (2016) suggested that the downstream grid cell ensemble performs principal component analysis (PCA) on the upstream place cell population activities. In so doing, a grid cell code is a compressed version of a place cell code via dimensionality reduction. Such a compression has been theoretically shown to be beneficial in the visual system (Riesenhuber and Poggio, 1999; DiCarlo and Cox, 2007; DiCarlo et al., 2012). However, in the case of spatial codes, a theoretical demonstration on its advantages is still lacking[1]. Regardless, I will discuss two major

[1]In the visual system, dimensionality reduction makes sense given the high-dimensionality of a raw image. In comparison, a spatial variable are very low-dimensional (1D or 2D). It therefore makes little sense to first encode using a high-dimensional representation (place cell code) and then again reduce back to a lower-dimensional representation (grid cell code).

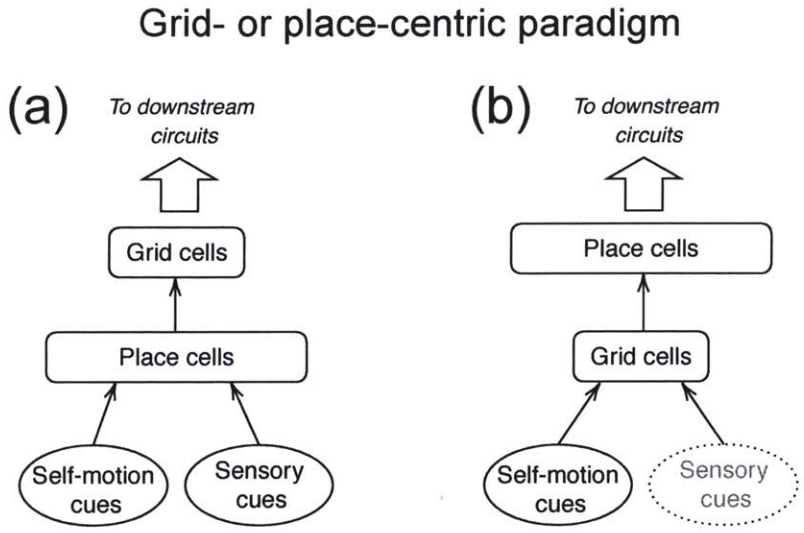

Figure 1.3: Grid cell or place cell centric paradigm. **(a)** Place-to-grid architecture with pre-existing place cells that transform sensory and self-motion cues to an allocentric position code and a feedforward projection, learnt via Hebbian rule, that performs principal component analysis (PCA) on the upstream place code (Dordek et al., 2016). Grid cell code can be seen as a compressed spatial code that might be useful for the downstream circuits **(b)** Grid-to-place architecture with pre-existing grid cells that path-integrate to provide an allocentric position code and a feedforward projection, learnt via Hebbian rule (Sreenivasan and Fiete, 2011), along with a winner-take-all place cell layer serving as a decoder. The decoded representation as a place cell code is easier to be interpreted for the downstream circuit.

predictions from this architecture that do not coincide with experimental observations.

First, although the feedforward projections in this architecture can be learned—with a modified hebbian rule and a non-negativity constraint—such that the downstream cells have triangular fields like grid cells, the model fails to produce modules (a major characteristic of grid cells) even after adding lateral connections within the output layer due to the fact that these cells were doing PCA largely independently.

Second, unlike grid cells, both the periodic tuning curves and cofiring structure would not be preserved after place cell remapping as illustrated in Figure 1.4. This example shows that a learnt feedforward projections in Environment 1 should have reminiscence in Environment 2 when the animal is first introduced to. Followed by a random shuffling of the place fields (global remapping), the original feedforward projections produce aperiodic tuning curves which was never observed in grid cells. And the change in the cofiring structure also contradicts the effect of grid realignment that otherwise preserves it as discussed in Section 1.1.1.

To resolve these contradictions, one may consider learning the recurrent connec-

tivity within output layer too (Widloski and Fiete, 2014) to build a stable continuous attractor that underlies periodic tuning curves. The issue is that such a remedy does not suggest any functional roles for this architecture either. On the other hand, there is another speculation regarding *vector navigation* (Bush et al., 2015; Banino et al., 2018; Baram et al., 2018) that do not regard a grid cell code to be a compressed spatial code. Instead, they hypothesized that the long-distance correlation from grid cell tuning curves is beneficial for performing vector navigation. However, a sound reasoning and demonstration are necessary before casting this speculation into a theory (more discussion on vector navigation in Chapter 2).

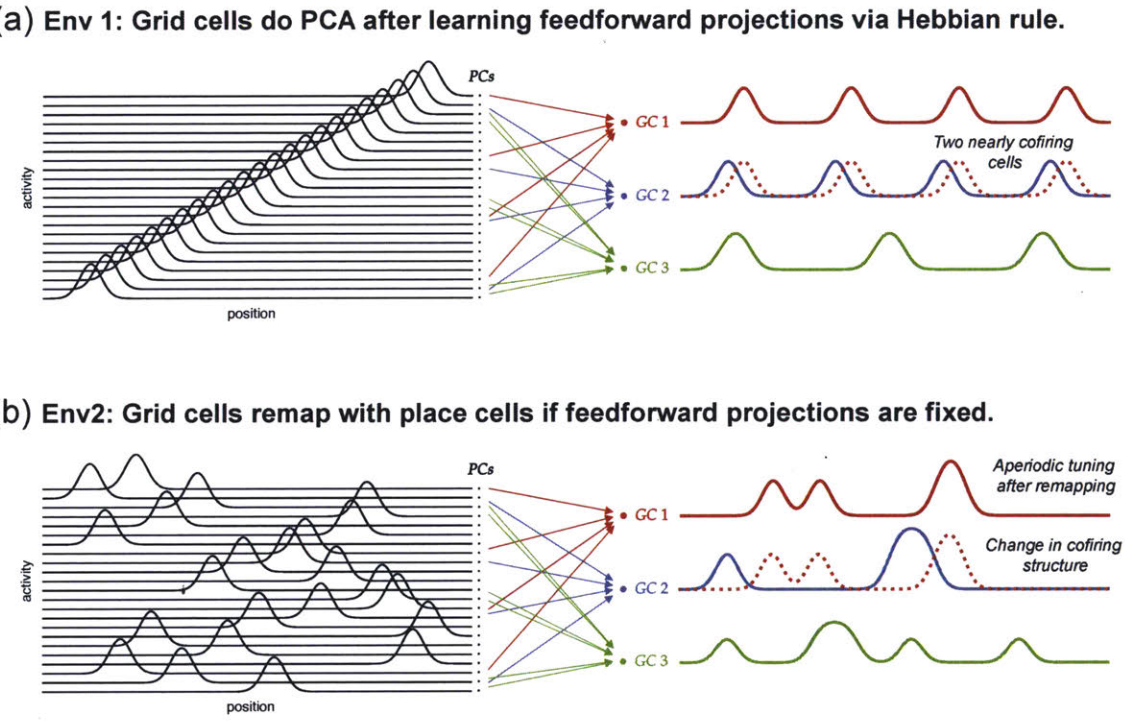

Figure 1.4: Periodic tuning curves and cofiring structure of grid cells are not preserved after place cell remapping in place-to-grid architecture. **(a)** The place cell tuning curves (black curves ordered by field location) in a 1D environment provide feedforward inputs to three example grid cells. The topographic feedforward connections learnt via Hebbian-type rule generate periodic tuning curves of grid cells that equivalently do principal component analysis on the place cell code (Dordek et al., 2016). Grid cell 1 and 2 are a slight shifted versions of each other so that they cofire with high probability. **(b)** With the previously learnt feedforward projections, the three grid cells lose their periodic tuning curves after the place cells remap. In addition, Grid cell 1 and 2 tuning curves have little correlations so that they rarely cofire in this new environment.

Grid-to-place architecture

The interpretation of grid-to-place architecture is relatively straightforward in which the place cell ensemble serves as a winner-take-all (WTA) decoder that transforms a complex dense representation of grid cells to a simpler sparse representation such that the animal's location can be directly read out (Sreenivasan and Fiete, 2011; Bush et al., 2015; Stemmler et al., 2015). However, this architecture directly contradicts some of the experimental results that agree with the place-to-grid architecture as mentioned above. But even putting those contradictions aside, there are still three predictions come out of this architecture that qualitatively do not agree with the experiments.

First, in experiments, place fields in a familiar environment remained intact after an inactivation of grid cells by either a specific removement of NMDA glutamate receptors (Gil et al., 2018) or by a disruption of theta oscillations (Brandon et al., 2014). The latter experiment also showed that a new set of distinct place fields can still form in a novel environment after grid cell inactivation. Yet the grid-to-place architecture predicts otherwise.

Second, in experiments, during the initial exploration of a novel environment, the place fields kept changing until finally stabilized after 10 mins (Wilson and McNaughton, 1993); in contrast, the fields were retained immediately when the animal was placed in a familiar environment. However, the grid-to-place architecture requires a relatively fixed decoder once it is learnt in order to maintain the same fields in a familiar environment; consequently the fixed decoder should immediately produce a new set of place fields in a novel environment without going through another learning process.

Finally, following the last reasoning, the subsequent sets of place fields should not be entirely uncorrelated across environments and exhibit a certain degree of regularity given that 1) there is only one decoder and 2) there is only one set of grid cells with highly regular activity patterns. At first glance, this prediction about an existent regularity in place cell tuning curves contradicts the experimental conclusion that the tuning curves across environments are highly uncorrelated. However, the signature of regularity could be small or hidden in some higher order statistics. With that, this potential regularity remains to be theoretically quantified and experimentally tested.

1.1.3 Function complementarity in grid-assist architecture

Grid-place complementary paradigm

From the discussion above, it is clear that neither of the two extreme architectures of grid- or place-centric paradigm can account for all experimental evidence. The lack of a specific theoretical motive also suggests a more reasonable paradigm that considers grid and place cell systems have complementary functions. There has been theoretical hypotheses that a grid cell ensemble with modular periodic tuning curves provides a topographic spatial code with metric information from which the distance and direction between any two locations can be precisely decoded (Moser et al., 2015), whereas a place cell ensemble with unimodal tuning curves does not provide a precise metric information but mostly encodes a graph that captures the spatial relationship among different locations, a property that defines topological codes (Chen et al., 2012, 2014; Curto, 2016; Curto et al., 2017; Dabaghian et al., 2012, 2014; Low et al., 2018)

There have been hints from either numerical simulations (Banino et al., 2018) or experiments(Morris et al., 1982; Gil et al., 2018) that either one of the two systems could be important depending on the tasks. However, it remains unclear under which circumstances a topological code is more advantageous than a topographic code, or vice versa. The question whether a dual topological-topographic code is necessary for a more general task needs to be answered as well. Having said that, it is without a doubt that the two systems are both essential for survival and one cannot simply be replaced by another. The grid-place complementary paradigm is thus a reasonable standpoint. Below, I will briefly discuss a more detailed architectural assumption for exploring this paradigm in the remaining chapters.

Grid-assist architecture

A grid-assist architecture illustrated in Figure 1.5 has the grid cell and place cell circuits operating independently. Both circuits process sensory and self-motion cues to give rise to their own distinct dynamics and tuning properties. And both provide the processed information to the downstream circuits. In this particular architecture, I consider a minimal interaction between the two circuits in which the grid cells provide sporadic feedforward inputs and supervised signal that assists place field formation during the learning phase in a new environment. After learning, in a familiar environment, place cell circuit goes back to operate on its own. Without

going into any functional reasons for this particular choice of architecture (which will be discussed in Chapter 3), it is worth mentioning that the grid-assist architecture is compatible with experimental results; that is, 1) the circuit underlies grid cells are largely fixed, whereas 2) the circuit underlies place cells changes on the demand through experience as discussed earlier in Sections 1.1.1 and 1.2.

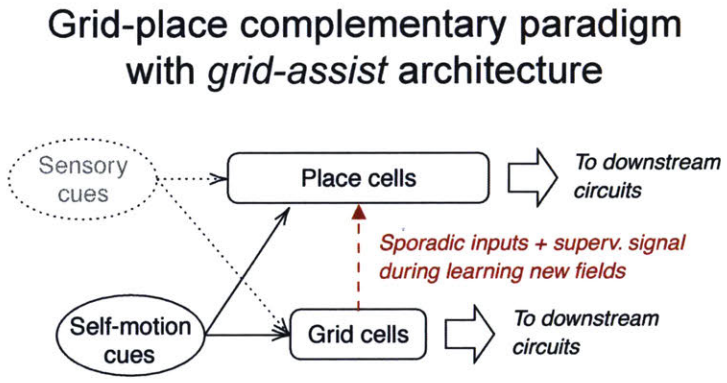

Figure 1.5: Grid and place cell complementary paradigm. In this paradigm, both grid cell and place cell representations have particular functions that are complementary to each other, and both representations are important for downstream circuits. A specific architecture called grid-assist architecture within this paradigm is adopted by this thesis. In which, the grid cells play an additional role that provide either a sporadic inputs or a supervising signal only during the period of place field formation in a novel environment. After learning, the two circuits operates independently. More details about this architecture will be discussed in Chapter 3.

1.2 Function hypothesis testing: Searching for the emergence

A function hypothesis in biology is a type of hypothesis that is lacking in other natural sciences like physics or chemistry. Going beyond establishing causation, It concerns purposes or functions for the existence of a given phenomenon (Wilson, 1991). In this section, I will first clarify the distinction between a function hypothesis and a more generic hypothesis in science. Then I will discuss how testing a function hypothesis requires a different approach then *modeling*, i.e., it requires to search for the emergence by demanding the system to perform hypothesized functions. At last, I will provide one example of the early visual system about how searching for the emergence can bring solid insights about its functions.

1.2.1 From phenomenology to function hypothesis

A generic hypothesis establishes a causal relation. Most of the time, it takes multiple cycles of hypothesizing, testing, re-hypothesizing, and re-testing before a well thought out hypothesis to be formalized. But in general, given an initial observation of a few related phenomena, a scientific hypothesis aims to find out causal relationships among them. For example, it has been observed that CA1 and CA3 place cells behave similarly (Leutgeb et al., 2004) to a large degree. Given the anatomical observations that 1) there are lots of connections between the two subregions, 2) there is no obvious recurrent connections among CA1 cells, and 3) there are lots of recurrent connections among CA3 cells (Witter and Amaral, 2004), one might hypothesize that *the stable CA1 place fields are caused by the feedforward inputs from the CA3 place cells.* Note that this hypothesis does not care about what functions either CA1 or CA3 or both subregions might perform, but only about establishing a causal relation (Krakauer et al., 2017).

A function hypothesis concerns purposive phenomena. When one talks about *functions* based on an observation, one implicitly assumes that a certain observed phenomenon exists for the purpose of having these functions. If finding out the causal relations among correlated phenomena is to answer the *how* question (how it works), finding out the functions underlies a phenomenon is to answer the *why* question (why it exists). The function, or purpose even, of a given neural phenomenon can be of either low- or high-level. A low level function could be a certain coding or dynamical property of a system that is important as the part of a higher-level goal. On the other hand, a high-level function can be directly related to a "goal-oriented" behavior, e.g., moving body in certain ways for minimizing energy costs, making a series of decisions for maximizing rewards, or perhaps just to simply play and explore.

Formulating a function hypothesis. A function hypothesis about the brain often started from relating the temporal sequence of a neural activity to an ongoing behavior. The construction of tuning curves is the most common first step: to find task or behaviorally relevant variables (e.g. position, speed, time, audio frequency, olfactory concentration, visual intensity, color, shape, etc.) such that the observed neurons selectively fire in response to them. Next, one may analyze an ensemble of cells with the same selectivity to relevant variables, and study whether such a population code is sufficient in solving the task before finally formalizing the function hypothesis. As an example, the observed orientation selectivity of V1 cells in the early visual system has been hypothesized as object edge filters for compressing the high-dimensional pixel

image on the retina before projecting to the downstream circuits (Daugman, 1988; Tai Sing Lee, 1996). In other words, one may hypothesize: *V1 orientation selectivity exists for a specific function of capturing edge information of an image.*

1.2.2 Experimental and theoretical frameworks on hypothesis testing

As discussed above, a function hypothesis is a statement goes beyond direct causation because it is not at all clear how the need for such a function can directly cause evolution to produce the corresponding phenomenon. Testing a function hypothesis, on this ground, is not as straightforward as testing a generic (causation-based) hypothesis. To a certain degree, one would need to imitate the process of evolution, or to seek for the emergence based on a hypothesized functions. Below I list a few experimental or theoretical frameworks for testing either a generic or a function hypothesis.

Testing a generic hypothesis: Phenomenon A causes Phenomenon B

In vivo experiment. First, one can probe the upstream system that produce Phenomenon A and see how Phenomenon B responds to this perturbation. Experimentalists often lesion or inactivate (e.g. genetic manipulation, drug, optogenetic perturbation) the upstream system to see if the downstream Phenomenon B is affected. Second, once the effect is confirmed, one may try to rescue the upstream system to see if Phenomenon B is restored. If yes, a direct causal link is established. If no, the inactivation of upstream system may cause other irreversible effects that subsequently affects Phenomenon B. The second rescue step is difficult to perform in early day lesion experiment but easier in nowadays optogenetics approach or the use of a reversible neurological drug.

Theoretical modeling. The advantage for a theoretical or numerical modeling is not only to reproduce the phenomena and understand its mechanism, but also to have an equivalent system where a more sophisticated perturbation or manipulation can be performed for establishing causation. For example, in the case of neural network modeling, individual change in synaptic connections can be made which is otherwise difficult in experiments. Moreover, given a potentially complex causal link between Phenomenon A and B, there could be more than one model that can reproduce the phenomenology. Each of such models could predict their own version of unforeseen

phenomena that might later guide the next experiment to discover some previously unseen Phenomenon C, and therefore establish a more complex and complete causal link.

Testing a function hypothesis: Phenomenon A exists for Function B

Searching for the emergence. Opposite to the modeling framework which builds a system that performs Function B from components that contain Phenomenon A, a framework that searches for the emergence does not hand-craft or prescribe the phenomenology. Instead, it finds *sufficient* conditions involving performing Function B that could lead to Phenomenon A. This framework is a weaker form compared to the optimization principle framework discussed next. Since this framework does not find the *necessary* conditions but only sufficient conditions, it's often hard to narrow down which subset of conditions might truly attribute to the existence of phenomenology.

Optimization principle. As the name suggests, an optimization principle framework involves defining an optimization problem. Specifically, it is to find a system to perform Function B and only Function B optimally. In other words, this optimal system emerges as a result of the necessary conditions for performing Function B. And if the optimal system also reproduce Phenomenon A, one has affirmed the function hypothesis in the strongest possible form. Below, I will use an example in the early visual system to explain how this framework could work before a more detailed discussion, in Section 1.3, for each step in applying this framework.

1.2.3 Emergence from optimizing functions affirms a function hypothesis: an example

The success in advancing the understanding of early visual system can be largely attributed to the application of optimization approaches. As an example, I will discuss below how optimizing a hypothesized function can lead to the emergence of orientation selectivity of V1 cells in the primate brain (Hubel and Wiesel, 1968). The function hypothesis stated:

> *V1 orientation selectivity exist to optimally recover the sources that underlie a natural image.*

It's worth noting that this hypothesized function were formulated normatively—i.e., based only on knowing that such a computation ought to be performed in the brain, and the computation has almost nothing to do with orientation selectivity at first glance.

Next, to quantify this function, Bell and Sejnowski (1997) assumed that the sources of an image have structures going beyond gaussian statistics. And since the resultant intensity of each pixel on the image is a mixture of multiple sources, its statistics appear to be more gaussian according to the *central limit theorem*. Having known that, one could argue that the intensity statistics from a source is maximally non-gaussian, so that their recovery means to maximize nongaussianity at the output layer—i.e., the V1 cells. Because the lowest order of cumulants for quantifying nongaussianity is *kurtosis*, the optimization objective is therefore to maximize the kurtosis of the intensity statistics from natural images.

Surprisingly, after the optimization using an efficient algorithm that performs an *independent component analysis*, the receptive field of a cell at the output layer is a Gabor filter with localized orientated shape resembling closely the receptive fields of V1 cells. This function hypothesis was therefore affirmed with a much stronger footing than those earlier studies that assumed and modeled only an edge-detection capability.

1.3 Optimization principle: a theoretical framework for function hypothesis testing

From the discussion and a working example in the above section, we have known that how the optimization principle framework can quantify and test a function hypothesis. In this section, I will start from motivating the reason behind doing *optimization* in studying a biological system. Next, I will define three types of constraints used in the framework and lay down specific steps which guide the search for an optimization principle. Finally, I will compare the optimization framework to well-known Marr's three levels of analysis that attempt to understand the brain by bridging the gap between animal's behaviors and neural phenomena at the circuit level.

1.3.1 Why optimization?

The concept of optimization has been around as long as the study of biology itself. Looking at nature carefully, it's not hard to see why this notion is so compelling. The evenly spiral arrangement of sunflower seeds utilizes the property of golden ratio for maximizing packing density (Mitchison, 1977); the prime 13- and 17-year lifespan of periodical cicadas keeps their genes that control lifespan intact by minimizing the probability of cross breeding such that these cicadas emerge all at once to survive through predator satiation (Cox and Carlton, 2003); or the single-photon detection in human vision is a result of the evolution pushing the light-detection sensitivity of photoreceptors up to their physical limit bounded merely by the fundamental neuronal noise at the molecular level (Bialek, 2012); the list can go on. Although we know that evolution solves a survival problem heuristically, the fact that optimal solutions abound in nature suggests that *natural selection* has imposed strong pressures towards individuals or groups such that they perform essential functions nearly optimally for survival within physical limits.

Even not knowing whether an optimal performance is truly required for a particular case, the principle of optimization in a sense still provides a useful way to "quantify" an otherwise subjective definition for "functions" in biology. To see this, consider the fact that there can be many different solutions underlie a single function each with its own distinct phenomenon. Given that the solution space is vast, the link between a certain phenomenon and a hypothesized function is weak by default. However, if one demands optimal performance on the function, it greatly narrows down the solution space as well as the possible phenomena these solutions can produce. Thereby, if the phenomenon, produced by one of the remaining solutions, matches the experimental observations, one can then make a strong claim that the observed phenomenon in experiment indeed exists for performing such a function.

1.3.2 From constraints to an optimization problem

Throughout the thesis I follow a guideline from the optimization principle framework as shown in Figure 1.6, in which a function hypothesis can be formulated into a precise optimization problem. Like mentioned, the rationale behind this framework is to see whether the optimal solution coincides with the targeted phenomenology. If so, one claims that the system emerges in the brain for performing such functions. If not, one is forced to revise the hypothesized functions given the fact that there are other

solutions without a given phenomenon that perform these functions better.

From functions to quantitative functional constraints. To formulate an optimization problem, a descriptive function needs to be quantified. It could be a measure of performance, e.g. fitness score, cumulative reward, or classification accuracy. It could also be the measure of a certain coding property, e.g. coding capacity, separability, decodability, or mutual information. Most of the time, a descriptive function, e.g. *navigation*, encompasses too many subcomponents and needs to be carefully broken down into a few separate functional constraints before formulating a corresponding optimization problem.

Formulating optimization problem with both functional and biological constraints. Besides functional constraints, biological constraints are equally important in the optimization principle framework, and one should not formulate an optimization problem without them. As we will see in the following chapters, certain phenomena could only emerge in the presence of a particular biological constraint. It would be mistaken to think of any salient phenomena exist for a functional need. An optimization problem thereby can be loosely defined as

$$\min_{\theta} \mathit{Loss}\,(\mathrm{fn1},\ \mathrm{fn2},\ \ldots \mid \mathrm{bio1},\ \mathrm{bio2},\ \ldots) \tag{1.1}$$

where the objective function, or the loss function, is a function of both functional or biological constraints which are parameterized according to an appropriate choice of *parameter space*. Without a well defined parameter space, both of these constraints cannot exist. It is natural to assume a parameter space specifies something physical—e.g. a neural network with synaptic connections to be optimizable parameters, but a parameter space can also specify something more abstract, like a set of tuning curves with an optimizable coefficient matrix (i.e. a coding theoretic approach to be discussed in Chapter 2).

Optimization, reformulating optimization problem, and revising hypothesis. After formulating an optimization problem, one needs to find its global optimum. For most of the problem in practice, this is an unsolved problem in computer science. The optimization methods adopted in this thesis are all based on stochastic gradient descent in which certain local minima may be avoided. For a multi-objective optimization problem, I developed a method called *stochastic orthogonal gradient descent* in order to reach *Pareto optimum* (Shoval et al., 2012), which will be discussed in Chapter 2. If one manages to reach global optimum (or close to it) but finds a mismatch in

phenomenology, a revision in the optimization problem is needed. One could revise certain biological constraints, reformulate optimization problem by choosing another system, or even revise the function hypothesis all together until the targeted phenomena emerges as the optimal solution.

Ensuring necessity of each constraint. After the appearance of targeted phenomena, one has shown a sufficient condition—i.e., the optimization problem with a particular set of constraints—for the emergence. To acquire an optimization principle, however, one needs to show such a sufficient condition is necessary as well. The step of ensuring necessity is less straightforward and rather depends on the format of an individual optimization problem. For an optimization problem loosely defined above in Equation (1.1), one could remove each constraint one at a time until a minimal set of necessary constraints has reached. After the necessity is ensured, an optimization principle is acquired, and the function hypothesis passes the test.

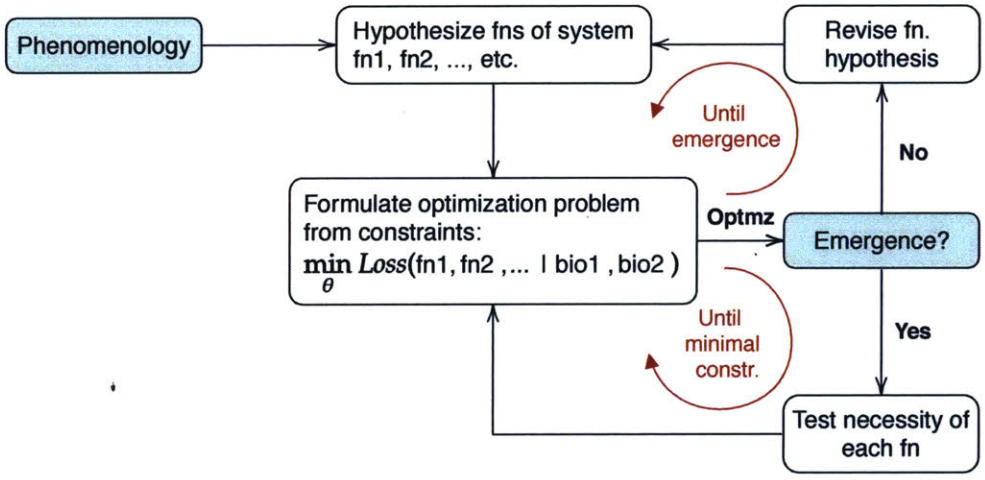

Figure 1.6: Optimization principle framework. 1) Starting from the phenomenology, one hypothesizes a set of functions the system performs. 2) Within a chosen parameter space, θ, that can describe the system, one formulate an optimization problem with a loss as a function of both functional and biological constraints. 3) If optimizing the parameters does not lead to the emergence of target phenomena, revise the function hypothesis, or reformulate the optimization problem until the emergence. 4) After the emergence, test the necessity of each constraint in the optimization problem until reaching a minimal set of necessary constraints.

1.3.3 Necessity of specific functional, biological, and core constraints

Necessity of high- and low-level functional constraints. Intuitively, a brain function could be either high-level that directly related to behavior or low-level that is part of a higher level goal. Because of this potential hierarchy, there is a notion of direct or indirect emergence. A direct emergence guarantees that the condition leading to it is necessary, whereas an indirect emergence does not. As an example illustrated in Figure 1.7 and explained in the caption, only low-level functions could lead to a direct emergence which reaches the criterion in the optimization framework. Therefore, as will be seen in Chapter 2 and 3, the hypothesized functions for place and grid cells are broken down to a few low-level functions in terms of coding or dynamical properties.

Biological constraints restrict a parameter space to biologically plausible regions. Similar to the philosophy of parameterization for formulating an optimization problem, biological constraints further narrow down the solution space as regions with biological plausibility. Similar to functional constraints, a different set of biological constraints terraforms an energy landscape such that the global optimum is relocated. In practice, a biological constraint can also take a *soft form* with a quantitative measure, e.g. energy consumption, to be optimized. In such a case, a biological constraint have equal footing as a functional constraint mathematically and does not directly limit search regions in parameter space with rigid boundaries. Consequently optimizing soft biological constraints could create a conflict with optimizing functional constraints and potentially leads to an emergence for the wrong reason—i.e., an emergence mainly comes from biological constraints such that the function hypothesis cannot be justified. It is thus important to keep in mind to assign a biological constraint when its really needed.

Core constraints is always necessary. Unlike a functional or biological constraint, a core constraint directly account for plausibility of the chosen parameter space for an optimization problem. The optimization principle framework I laid out demands the fulfillment of core constraints. In other words, one cannot declare an optimization principle if the optimal solution is not realizable in the brain even after the emergence. For example, in a coding theoretic approach, an optimal solution takes the form of tuning curves; and it is necessary to find a corresponding neural implementation for the optimization principle to be finalized.

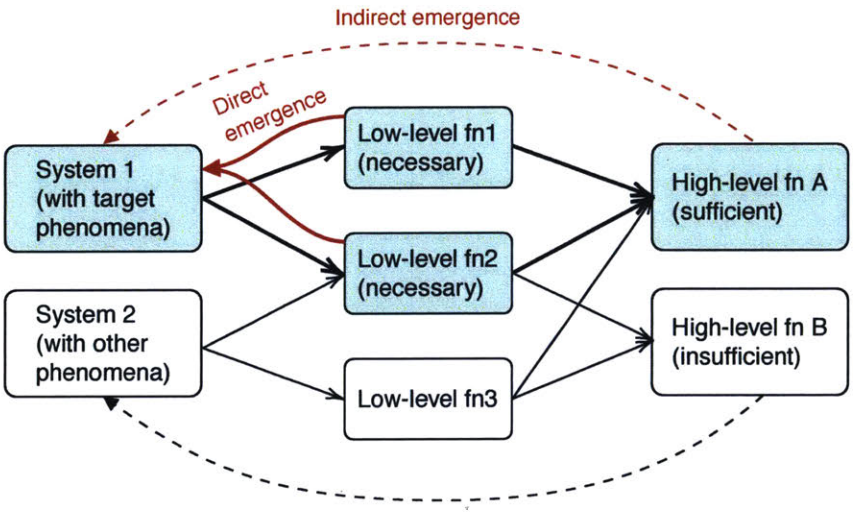

Figure 1.7: Notion of direct or indirect emergence. System 1 with the targeted phenomena simultaneously performs both low-level Function 1 and 2 optimally. If an optimization problem is formulated at this level, the optimal solution will be System 1 as a result of *direct emergence*. However, if an optimization problem is formulated based on a high-level Function A which consists of all three low-level functions, an *indirect emergence* could happen with the optimal solution exhibits characteristics of both System 1 and 2. An indirect emergence is therefore a result of only a sufficient condition, not a necessary one, and thus it does not match the criterion in the optimization principle framework.

1.3.4 In comparison with Marr's three levels of analysis

Given that the optimization principle framework connects the circuit-level neural phenomena to the higher-level functions, we can compare it with Marr's three levels of analysis (Marr and Poggio, 1976) (also see (Krakauer et al., 2017) for a comprehensive review) and see what in addition this framework may offer. As illustrated in Figure 1.8(a), Marr's three levels start from defining a *computation problem* that is essential for the brain to solve. Next, the effort goes into finding an *algorithmic solution* that can realistically solve the problem before eventually implement such an algorithm in a physical system—e.g. the brain. In Marr's framework, each level is a successive realization step of the previous level following a relatively strict hierarchy. However, the three analogous levels in the optimization principle framework illustrated in Figure 1.8(b) are not strictly hierarchical. In this framework, both the top and bottom levels have an influence on the middle level. The top level demands not only "a" solution but "the" solution that is optimal at the algorithmic level. On the other hand, the bottom level imposes another constraint (the core constraint) that demands a plausible neural implementation. Having the demands from both levels, the solution space at the algorithmic level is greatly reduced, and thus more likely to

be the only solutions left that nature might have discovered in the brain.

Another potential advantage with these non-hierarchical levels comes from an equal emphasis on both the algorithmic and implementation level. In Marr's unidirectional realization steps, one naturally attempt to find a working algorithm with the least parameters (a minimal model) based on the computation problem. However, even the found solution appears conceptually simple at the algorithmic level, it might not be at the implementation level. These hierarchical steps thereby create a bias towards finding a solution that fits human intuition instead of discovering a solution adopted by the brain. As an example, many algorithms used to model vision did not concern dynamics at all because of this bias. In contrast, the optimization principle framework finds a solution that is optimal at both algorithmic and implementation levels. Because there is no inherent bias to find a minimal algorithm, this framework could 1) lead to novel insights about how the observed phenomena might come about due to the dynamical constraints at the implementation level, or 2) lead to a discovery of an counter-intuitive algorithmic solution that is otherwise plausible to be implemented in the brain. More recently, in similar philosophy, Poggio (2012) revisited Marr's three levels of analysis and argued that, to better understand the brain, one should put more emphasis on the learning dynamics as well.

1.4 Structure of the thesis

The rest of the thesis applies the framework of optimization principle on grid and place cells separately.

Chapter 2 starts from hypothesizing grid cell functions based on their major tuning properties and their lack of slow dynamics. Next, an RNN training approach is applied to encompass only the hypothesized low-level functions. In which, I will compare my training results to Deepmind's and reason why the training failed to converge, and what the implications are regarding the grid cell circuit. From there, a pure coding theoretic approach is developed to circumvent the issues from the RNN training approach. And it is followed by a journey of formulating and reformulating the optimization problem with three major optimization schemes each of which aiming for resolving the issues of the last.

Chapter 3 starts from, again, hypothesizing functions but for place cells. Then, I will introduce an *optimal online learning algorithm* as the first half of the optimization

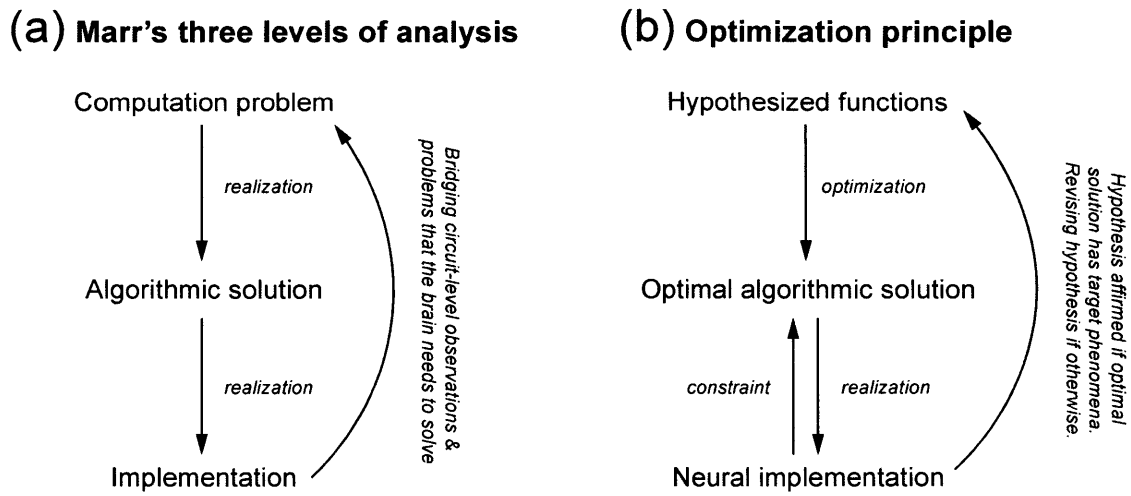

Figure 1.8: Optimization principle frameworks vs. Marr's three levels of analysis. **(a)** The unidirectional hierarchy among three levels inherently create a bias towards finding minimal algorithmic solution for a given computation problem. The conceptually easy solution at algorithmic level might not be simple or plausible at implementation level. **(b)** The algorithmic level—having constraints from both top and bottom levels—finds optimal solutions for the hypothesized functions that can be implemented in the brain. Without putting particular emphasis that prioritize the algorithmic level, the optimization principle framework provides a guideline for discovering a counter-intuitive but otherwise plausible solution in the brain.

principle that is compatible with finding a neural implementation. Next, I will use a simplified task to investigate the online learnability of an RNN in order to find a plausible neural implementation. In which, I compare this learnable RNN to an existing *multichart attractor network* model for place cells, discuss how the new model resolves the implausibility of the old model, and make experimental predictions. At last, I will illustrate how the optimal algorithm can be implemented based on the result from the learnable RNN, and, from there, predict the origin of the biased field propensity observed in the CA1 place cells.

Chapter 4 concludes all the important insights discovered in this work.

Chapter 2

Grid cells emerge as an optimal high-capacity robust code

2.1 Introduction

In Chapter 1, we were introduced to the notion of function complementarity in a grid-assist circuit architecture. It is a fundamental stance of this thesis that regards place and grid cells as a dual system with different sets of functions that are complementary to each other. Although this architecture has the two systems as a whole, they are also largely independent. In it, grid cells are not just to assist place cells for learning a novel environment; they also have distinct functional roles and can operate in isolation. Therefore, it is possible to independently formulate a function hypothesis for grid cells. And for that, the purpose of Chapter 2 is to focus on testing grid cell function hypotheses alone via different approaches and optimization schemes.

2.1.1 Function hypothesis: High capacity robust coding

In Chapter 1, we were introduced to the tuning properties and circuitry of grid cells. Here, we will start discussing their potential functions and formulate a plausible function hypothesis for later testing via optimization principle approach. A function can be of low-level or high-level. In the context of grid cells, a low-level function is directly hypothesized as a coding property derived from their tuning properties; whereas a high-level function can be a combination of a few low-level functions or something totally new without clear connection to the low-level.

A low-level function is a coding property

Recall that there were three essential tuning properties introduced that defines a population of grid cells:

1. They have spatially periodic tuning curves (Hafting et al., 2005).
2. They form a few discrete modules of different periods (Stensola et al., 2012).
3. Within an individual module, there are uniform phase coverage (Hafting et al., 2005; Stensola et al., 2012).

A low-level function as a corresponding coding property can be directly derived from these tuning properties. Fiete et al. (2008) provided theoretically derived coding properties including:

1. Exponential capacity with respect to the number of cells.
2. Efficient serial update of codewords via parallel modular addition. coverage (Hafting et al., 2005; Stensola et al., 2012).

The first property regarding coding capacity was derived from the first and second tuning properties, i.e. a set of modular periodic tuning curves. The derivation followed directly from modular arithmetic as will be discussed in detail in Section 2.3 when we talk about binary grid cell code. The second coding property was derived from all three tuning properties. It describes an efficient codeword progression that implements a series of updates in animal's location. For this reason, it directly relates to path-integration. In addition, this coding property can take another format rendering a code translationally invariant (TI) as its importance will be further discussed in Section 2.3.

A type-i high-level function directly combines coding properties

There exists functions at higher level that is not immediately obvious until one combines few low-level ones. As an example, a grid code has an extra singularly precise error-correction capability as a result of combining the above two coding properties (Sreenivasan and Fiete, 2011). To understand how this higher function arises, imagine a code with a set of modular periodic tuning curves. The global geometric structure

of its coding line (assuming a 1D spatial code) will be arranged in a nested fashion. That is, nearby locations are encoded on the same augmented long segment of coding line, and distant locations are encoded in parallel segments that are packed closely to save space. In such a coding line arrangement, the posterior of a decoded location is redistributed leaving a singularly precise sharp peak at true location.

A type-ii high-level function indirectly relates to the coding properties

Other grid cell functions were also proposed that do not have an obvious connection to the coding properties.

Vector navigation. From observation of extended tuning curves, grid cells have been hypothesized to perform vector navigation (Bush et al., 2015; Stemmler et al., 2015; Banino et al., 2018) in which an animal is capable of figuring out a goal-vector—i.e. the direction and distance of a goal-location from current location—using merely the codewords of goal and current location. The two type of goal-vector decoding processes were studied:

1. Direct goal-vector decoding (Bush et al., 2015; Stemmler et al., 2015)
2. Linear-look-ahead goal-vector decoding (Erdem and Hasselmo, 2014, 2012; Kubie and Fenton, 2012)

Both of the above compute a goal-vector as necessary information for vector navigation. That said, however, it is also possible to perform a similar vector navigation without explicitly computing a goal-vector as we will discuss in Section 2.2.3. The process is related to multi-policy successor representation (SR) in a reinforcement learning (RL) task. Multi-policy SR (Barreto et al., 2016) or simply SR (Stachenfeld et al., 2017) in spatial RL tasks have been studied extensively. In those studies, grid cells' peculiar spatially extended tuning curves have been thought to play an important role in efficiently and effectively building SRs.

Computation in conceptual space. There are also speculations and theories regarding the functions of grid cells beyond spatial tasks (not the scope of the thesis) which I list a few below.

1. Better computation in conceptual space (Herz et al., 2017; Behrens et al., 2018)
2. Generalisation of structural knowledge (Whittington et al., 2018)

Taken together, we will start pursuing an optimization principle from considering the two low-level functions. A tentative function hypothesis can be stated as:

Grid cells exist for having a high-capacity and robust path-integrating code.

We will see how this hypothesis stands as we walk the journey of testing the necessity of each function in various optimization schemes.

2.1.2 Optimization principle: Focus on searching necessary functional constraints

To test the function hypothesis, we need to further formalize these low-level functions—or coding properties—as a set of functional and biological constraints in precise mathematical forms. E.g. formulating a set of loss functions in an optimization problem. But before doing so, we must remember—back in Chapter 1—to fulfill the non-negotiable core constraint demanded by optimization principle approach. That is, to demonstrate neural implementation of the corresponding code.

Demonstration of neural implementation in this thesis can be twofold: 1) to show the existence of a circuit and 2) to show the existence of learning rules that wires up the circuit. In the context of grid cells, showing the existence of learning rules is not a concern. The reason is that—unlike a place cell circuit—a grid cell circuit, to a large degree, does not seem to change much across behavioral timescale. Circuit learning therefore is irrelevant to grid cell functions. And for this reason, we only need to demonstrate the existence of circuit. This realization gives us a big freedom in designing optimization scheme, because—as long as we can—it doesn't matter how we get to the optimal solutions. And for that, we will be carrying out two different optimization schemes: RNN training scheme and coding theoretic optimization scheme.

In the second part of Chapter 2 where we discuss coding theoretic approach, the focus will be entirely on searching necessary functional and biological constraints on the code without worrying about underlying circuitry that generates the code. It is justifiable to do so because it has been demonstrated that for each plausible grid cell code, there exists a corresponding RNN solution (Burak and Fiete, 2009). More detail about moving towards coding theoretic approach will be discussed in Section 2.2.

2.1.3 Chapter organisation

The rest of Chapter 2 has two parts. Part 1, or Section 2.2, focuses on approach of RNN training—which fulfills core constraint automatically. At the end of this section, I will argue—based on both my training results and Deepmind's—why it makes sense moving from an RNN training to a coding theoretic approach. Part 2, spanning from Section 2.3 to the end, focuses on coding theoretic approach. It is a journey of formulating, testing, and reformulating details in constraints regarding the function hypotheses. At the end of Chapter 2, I will summarize the insights acquired and—based on those—where we may go next for future study.

2.2 From training RNN to coding theoretic approach

In this section, we will start from an approach of training recurrent neural network (RNN). I will demonstrate the result from two different schemes aiming to implement a task that directly demands two main grid cell functions from our earlier function hypothesis—*Grid cells exist for having a high-capacity and robust path-integrating code.* I will explain why both training schemes fail to converge and compared them to a recent study from Deepmind (Banino et al., 2018) on the same topic.

I will reason why Deepmind's training scheme was not yet based on a concrete function hypothesis so that the scheme—leading to observation of grid cell formation—cannot account for an optimization principle. I will discuss their demonstration of vector navigation—an actual function hypothesis they had in mind—via a reinforcement learning (RL) task. I will then proceed to give an alternative explanation regarding their vector navigation result, and formulate a corresponding high-level function hypothesis based on this new explanation.

At last, I will end the section by motivating another entirely different kind of approach for pursuing an optimization principle, i.e. a coding theoretic approach. I will point out its advantages and legitimacy regarding removing RNN dynamics.

2.2.1 Training RNNs: Grid cells emerge to perform a task

An approach of RNN training lies on the edge for calling it an approach of optimization principle. In general, it is—at most—an approach at the level of *searching for emergence* as we discussed in Chapter 1. In an RNN training, we demand the agent

to perform a task—which could be thought as a high-level function—that indirectly combines few low-level functions (or coding properties in context of grid cells). Because this approach does not target low-level functions, the emergence it provides—if any—is a result from a combination of all necessary and possibly some unnecessary constraints. Very often, a designed task inevitably encompasses too many unnecessary constraints including those of coding, dynamical, or other uncategorizable properties of RNN. Because of that, one can only claim sufficiency after emergence at best[1].

Having said that, training RNN remains appealing and should be our starting point for that it is the most straightforward setup to translate a function hypothesis into precise mathematical form. And for that, most importantly, it automatically fulfill the core constraint—need for a neural implementation—in optimization principle approach (see Chapter 1 for details). With a careful craft on designed task and RNN architecture, one may still claim necessity—not just sufficiency—of a task for emergence. Next, I will show the setups and results from two RNN training schemes, and discuss how they fit into the criteria of being an optimization principle approach.

2.2.2 Two RNN training schemes aiming for an optimization principle

To truly make an approach of training RNN an approach of optimization principle, one needs to clearly separate undesirable constraints of RNN dynamics (and even learning dynamics) from necessary functional constraints of a task[2]. It is however inherently impossible to do so, because *a task is performed by an RNN*. Consequently, dynamical and functional constraints are inevitably tangled in an RNN. As a starting point, one may keep RNN architecture as simple as possible such that an emergent phenomenon can be unambiguously attributed to either functional or dynamical constraints. Also, one should keep task straightforward such that it directly connects to the demanded coding properties (hypothesized low-level functions).

The tasks behind my two training schemes are the same. They both demand an RNN agent to accurately self-localize in a large environment. The corresponding loss function for the task is simply a square error of estimated position. To succeed in this task—which is a high-level function, an RNN agent needs and only needs the

[1]Another different task might also cover all necessary constraints attributed to the emergence. For that, one can, at most, claim sufficiency for these tasks leading to emergence.

[2]One cannot claim optimization principle if a dynamical constraint ends up leading to emergence, because it is not in the function hypothesis.

two low-level coding properties in our function hypothesis: 1) high-capacity encoding and 2) accurate path-integration. For this reason, these training schemes are aiming for an optimization principle.

Note that, since the loss function assigns none of the two coding properties, these properties can only be implicitly specified in details of a training environment and an RNN architecture. Both Scheme 1 and 2 share the same training environment. The difference between the two lie in their architectures as we will discuss next.

Setup of Scheme 1

The environment and architecture of Scheme 1 is shown in Figure 2.1(a,b). For RNN agent to acquire path-integration ability—i.e. integrating self-motion cue to self-location—the landmarks in the environment should be sporadic and only serve for the purpose of an occasional error correction. I therefore used dilute landmarks as infrequent as one landmark per meter or every 5 seconds (with an average speed of 20 cm/s). This setup forces an RNN to generate a persistent activity pattern—instead of simply passing landmark input in a feedforward way to the output—over few seconds for continuously representing positions between two adjacent landmarks where there is no external input.

As for acquiring high-capacity encoding, I used a 50-meter-long track. To judge the relative size of this coding range, we need to compare it to the number of cells used in the main computation layer, i.e. the grid cell recurrent layer s_G in Figure 2.1(b). I used a heuristic estimate[3] on coding range in unit of number of cells:

$$\ell \approx \frac{L}{\overline{v} \cdot 1s} = \frac{50m}{25cm} = 200 \geqslant 100 \text{ (or } 200) = N_G \tag{2.1}$$

The number of cells is chosen such that the coding range is large in all my training schemes. With that, a low-capacity type of encoding (e.g. place cells) is unable to

[3]A precise estimate on coding range will need to take RNN dynamics into account. The problem is challenging because of the unpredictable nature of RNN dynamics—given that there are virtually infinite many of them—in response to a noise source. An attempt to quantify the effect on a well-structured continuous attractor network has been made: (Burak and Fiete, 2012). However, much less is known if the RNN is also highly non-autonomous—receiving landmark and velocity inputs—or has a non-canonical recurrent dynamics, e.g. a LSTM adopted by Deepmind as will be discussed next. Also, in practice, to reduce the computation cost in training, the simulation timestep cannot be too small. This discreteness on the otherwise smooth activities and inputs could cause nonphysical stochasticity mixed with a physical one. To conclude: If there is no well defined stochasticity in RNN as a yardstick, there is no sense of counting how many separable states an RNN can produce—i.e. no reliable way to estimate a coding range.

solve the task and will not emerge as an optimal solution.

In this scheme, both grid and place cell layers are recurrent networks. The recurrent, feedforward projection, and feedback projection weights of grid cell layer is trainable. The dynamics of grid cells evolve according to

$$\mathbf{s}_\mathbf{G}^{t+1} = (1-\eta)\ \mathbf{s}_\mathbf{G}^{t} + \eta\ \sigma\left(U_v x_v^t + W_{GG}\mathbf{s}_G^t + W_{GP}\mathbf{s}_P^t + b_G + a_\xi \xi^t\right), \tag{2.2}$$

where $\sigma(u) = [1 + \exp(-4(x - 1/2))]^{-1}$ is a scaled-shifted sigmoid function that would promote dense activity of grid cells. The velocity input x_v^t are generated via a correlated random walk. And a weak independent gaussian noise $\xi^t \sim \mathcal{G}(0,1)$ is added for each cells. The place cell layer is a fixed winner-take-all (WTA) recurrent network. The dynamics of place cells evolve according to

$$\mathbf{s}_P^{t+1} = (1-\eta)\ \mathbf{s}_P^{t} + \eta\ \left[U_l \mathbf{x}_l^t + W_{PP}\mathbf{s}_P^t + W_{PG}\mathbf{s}_G^t + b_P\right]_+, \tag{2.3}$$

where $[u]_+ = \begin{cases} 0 & \text{if } u < 0 \\ 1 & \text{if } u \geqslant 0 \end{cases}$. And a landmark input x_l^t is only active when the agent is near landmark l (see Appendix A.8 for detail). The target and batch loss function are given as

$$\mathbf{y}^t = U_l\ \mathbf{x}_l^t, \tag{2.4}$$

$$\text{Batch loss} = \frac{1}{2}\sum_{t=t_0}^{t_0+T}\left[\left(\mathbf{y}^t - \mathbf{s}_P^t\right)^2\ *\ \left[\sum_{i=1}^{N_l}\mathbf{x}_l^t - .9\right]_+\right], \tag{2.5}$$

where $\left[\sum_{i=1}^{N_l}\mathbf{x}_l^t - .9\right]_+$ is a mask for only activating gradient descent update when the agent is very close to one of the landmarks. Each training session comes with a batch of 10 parallel trials. To minimize loss, backpropagation thought time (BPTT) algorithm was used with a-second-long backprop time window.

To succeed the task, place cell state $\mathbf{s}_p$ cannot simply pass on the landmark input $U_l\mathbf{x}_l$ in real time to its own current activity; i.e. Equation (2.3) would not be satisfied because of the presence of WTA recurrent dynamics. Rather, it is important that place cells' last state, its recurrence, and grid cell inputs all work together to generate coherent temporal dynamics for matching the target.

Results of Scheme A is shown in Figure 2.2. The trained RNN failed to accurately estimate position—which is measured by running average of Batch loss, and the

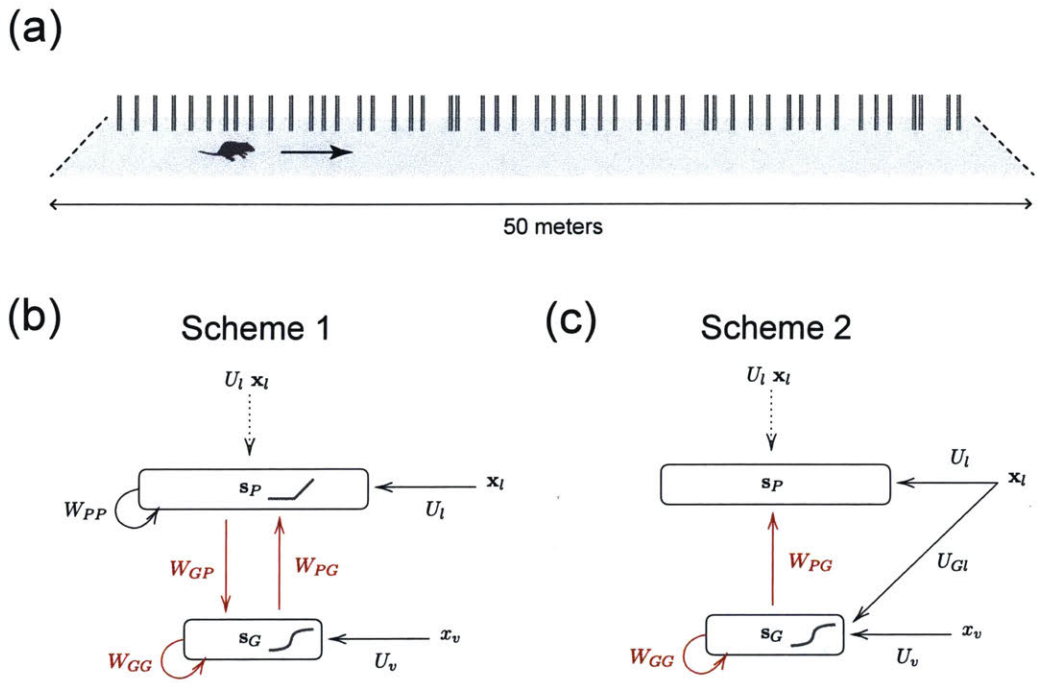

Figure 2.1: RNN training schemes in a large 1D environment. **(a)** The agent random walked and is demanded to self-localize in a 50-meter-long 1D circular track. **(b)** Network architecture of Scheme 1 had an essential RNN as grid cell layer s_G that integrated velocity input x_v, and a WTA place cell layer as a decoder of upstream grid cell code for consistently matching the targeted landmark signals x_l. **(c)** Architecture of Scheme 2 that simplified place cell layer by removing its dynamics. In addition, the grid cell layer also received a landmark input x_l for a more stabilized training process. All red-marked weights are trainable.

trainable weights did not converge—which is measured by the gradient

$$\Delta W_{GG} \propto \frac{\partial \text{ Batch loss}}{\partial W_{GG}}$$

as shown in Figure 2.2(a). The final loss is roughly 6 times lower than the initial, this is really not remarkable compared to the result from a similar training scheme but with a far smaller final loss—the loss is roughly 100 times lower than the initial, or the error in position estimate is 10 times lower than the initial (keep in mind that—in that training to be discussed in Chapter 3—the environment is also smaller: $L \approx 10m$).

It is worth pointing out that, in this result, I did see a sign of periodic tuning curves in the grid cell layer as shown in the enlarged area of Figure 2.2(b), however, as one might expect, this activity was only transient and kept changing from epochs to epochs because the non-converging weights. At the time, I speculated the unsuccessful

training can be attributed to the slow WTA dynamics of the place cell decoder. The slow dynamics could cause an unstable delay in transmitting grid cell activity pattern to the final output, and ultimately cause the change in weights never settling down. For this reason, I removed the dynamics of place cell decoder in the next scheme.

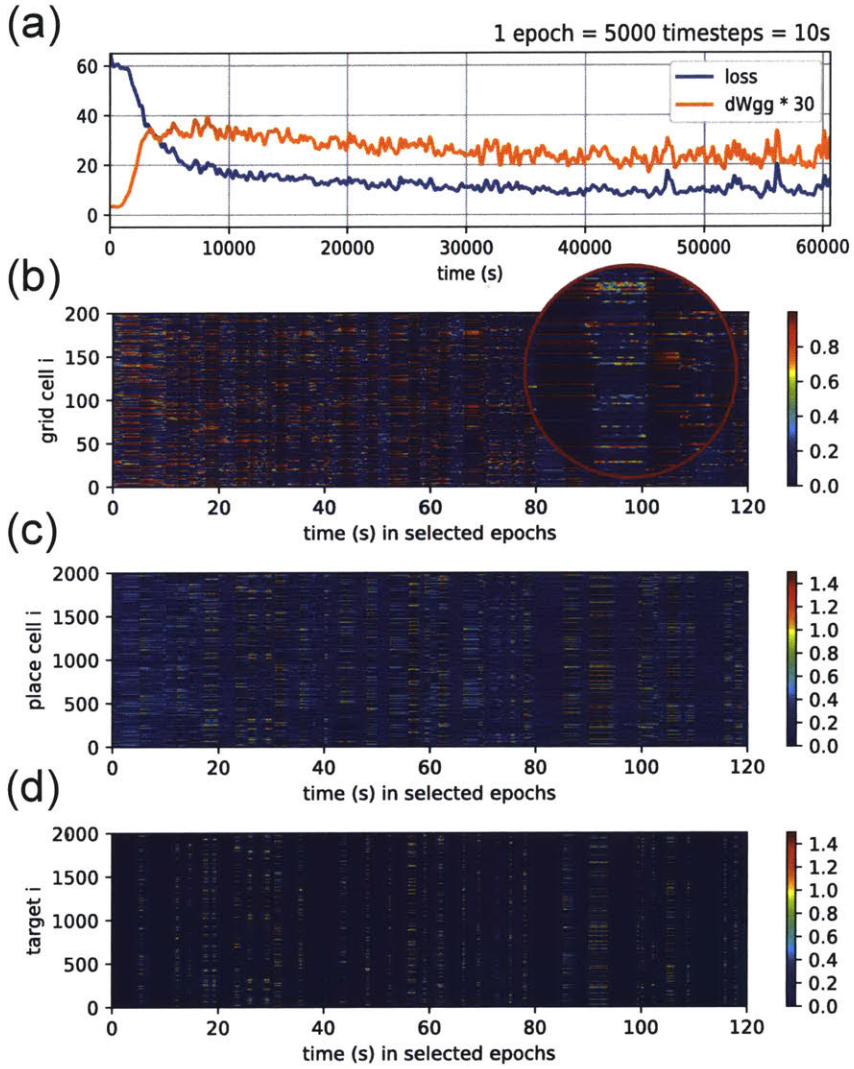

Figure 2.2: Learning did not converge in Scheme 1. **(a)** Neither loss as landmark signal estimation error (blue) nor recurrent weight change (orange) converge for a successful learning. **(b)** Grid cell activity in selected epochs. Oscillatory activities started to become visible (enlarged) toward later epochs, but they were not stable. **(c,d)** place cell activities and their corresponding targets.

Setup of Scheme 2

In this scheme as shown in Figure 2.1(c), only grid cells has recurrent dynamics. Their recurrent and feedforward projection weights are the only two trainable parameters.

The recurrent dynamics evolve according to

$$\mathbf{s}_{\mathbf{G}}^{t+1} = (1-\eta)\ \mathbf{s}_{\mathbf{G}}^{t} + \eta\ \sigma\left(U_v x_v^t + U_{Gl}\mathbf{x}_l^t + W_{GG}\mathbf{s}_G^t + W_{GP}\mathbf{s}_P^t + b_G + a_\xi \xi^t\right), \tag{2.6}$$

here $\sigma(u) = [1+\exp(-4(x-1/2))]^{-1}$is a scaled-shifted sigmoid function that would promote dense activity of grid cells. The velocity input x_v^t are generated via a correlated random walk. A landmark input x_l^t is only active when the agent is near landmark l (see Appendix A.9 for detail). And a weak independent gaussian noise $\xi^t \sim \mathcal{G}(0,1)$ is added for each cells. The place cell layer is now a simple softmax function that nonlinearly transform the linear sum of grid cell activities. The output is given as

$$\mathbf{s}_P^{t+1} = \text{softmax}\left(1/a_l\ U_l\mathbf{x}_l^t + W_{PG}\mathbf{s}_G^t\right), \tag{2.7}$$

The target and batch loss function are given as

$$\mathbf{y}^t = U_l\ \mathbf{x}_l^t/(.1N_P), \tag{2.8}$$

$$\text{Batch loss} = \frac{1}{2}\sum_{t=t_0}^{t_0+T}\left[\left(\mathbf{y}^t - \mathbf{s}_P^t\right)^2\ *\ \left[\sum_{i=1}^{N_l}\mathbf{x}_l^t - .9\right]_+\right], \tag{2.9}$$

where $\left[\sum_{i=1}^{N_l}\mathbf{x}_l^t - .9\right]_+$ is a mask for only turning on gradient descent when very close to one of the landmarks. Each training session comes with a batch of 10 parallel trials. The loss batch loss function is minimized via BPTT algorithm with a-second-long backprop time window.

To succeed the task, place cell state $\mathbf{s}_p$ cannot simply pass on the landmark input $U_l\mathbf{x}_l$ in real time to its own output, because the mismatch caused by softmax' nonlinear transformation of the input. Rather, it is important to include grid cell inputs for compensating the mismatch to the target.

Result of Scheme 2 is shown in Figure 2.3. The major takeaways from this scheme is the same as that from Scheme 1. The RNN failed to accurately estimate position, the weight changes did not converge, and while there was a sign of oscillatory activities for spatially periodic tuning, they were transient and kept changing from epochs to epochs.

After many attempts and hyperparameter fine-tuning and after comparing to Deepmind's result—we will discuss next, I concluded that the schemes I set up are

too challenging such that the high-performance solutions—potentially grid cell solutions—are rare and unreachable via a gradient-based optimization. These results further motivated a coding-theoretic optimization approach that may have better chance to reach these islands of solutions, which we will discuss at the end of this section.

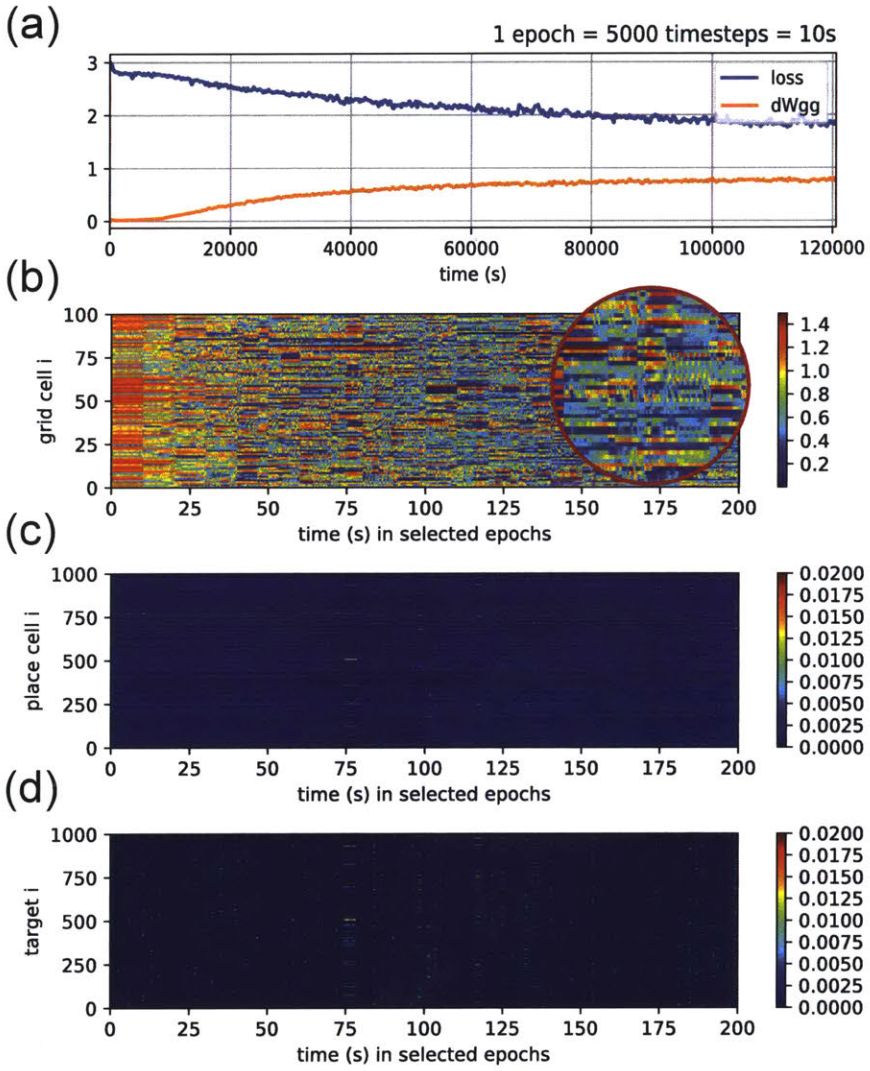

Figure 2.3: Learning did not converge in Scheme 2. **(a)** Neither loss as landmark signal estimation error (blue) nor recurrent weight change (orange) converge for a successful learning. **(b)** Grid cell activity in selected epochs. Oscillatory activities started to become visible (enlarged) toward later epochs, but they were not stable. **(c,d)** place cell activities and their corresponding targets.

2.2.3 Deepmind's training scheme & general takeaway

In this section, we will discuss a result from deepmind's recent paper (Banino et al., 2018). I would also like to point out a similar attempt has been studied in (Cueva and Wei, 2018). The training scheme and RNN architecture from Deepmind are shown in Figure 2.4. At the end of training, they were able to acquire grid cell tuning curves for 20% of cells in a dropout layer. So, why can their training be successful whereas mine failed to converge? what is the mechanism in their scheme that drives grid cell formation? and what does their results speak about function hypotheses and optimization principle? By the end of this discussion, I hope I can provide some insight into these questions.

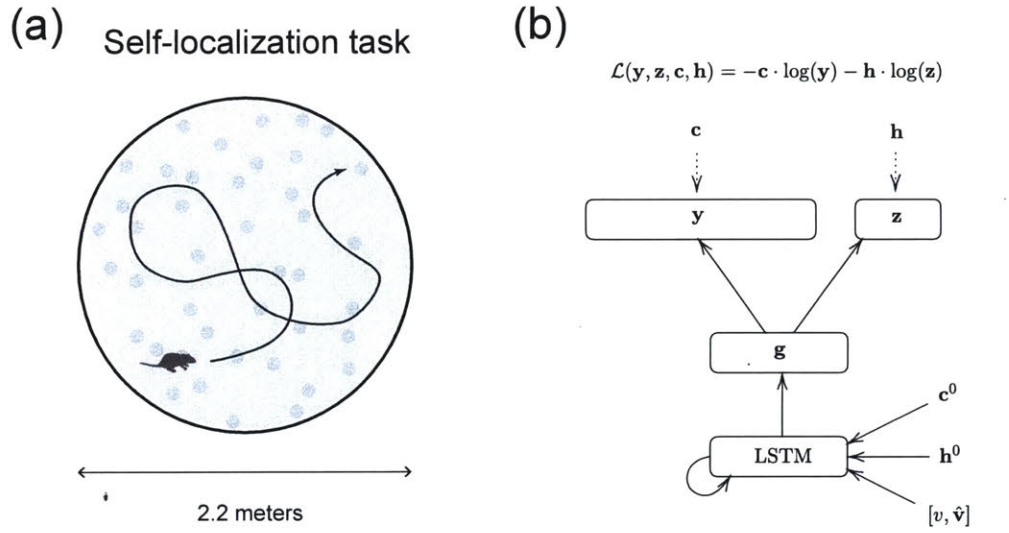

Figure 2.4: Deepmind's supervised learning schemes in a small 2D environment. **(a)** The agent random walked and demanded to self-localize in a small circular arena with initial place cell inputs acting as landmark—depicted as background gray points—corrections. **(b)** Network architecture with an essential LSTM for path integration and a dropout layer g for regularization that responsible for the emergence of grid cells. c and h are ground-true head direction and place signals.

Not aiming for high-capacity encoding

Deepmind had two training schemes. The first one, seeking for grid cell emergence, is a supervised self-localization task in a small square or circular environment (The second reinforcement learning (RL) scheme will be brought up later). The size of the environment is $2.2m \times 2.2m$. If using unit of number of cells from our heuristic estimate above, the corresponding coding range is marginally smaller than the number

of LSTM, short for long short-term memory, units:

$$\ell \approx \left(\frac{L}{v}\right)^2 = \left(\frac{2.2}{1.5}\right)^2 = 215 < 256 = N_{LSTM}$$

Because of that, a low-capacity code of place cell type is possible to solve the task. In fact, in the Extended Data of the paper, they did find the solution to be of place cell type when turning off the dropout regularizer as discussed next.

An additional dropout regularizer and a conventional wisdom in deep learning

From Figure 2.4(b), we note that the LSTM layer does not directly project to the output decoder y and z, even though a direct projection would be the most sensible choice given that LSTM contains all spatial information about agent's position. Instead, Deepmind inserted an additional dropout layer g in between with 50% of randomly selected cells being turned off at any instances. In the discussion below, I would like to leave the argument about biological plausibility of this dropout layer behind, and just try to understand the effect and meaning of such architecture.

First, the dropout layer contains 512 cells. It therefore has more than enough capability to preserve all the information from the upstream LSTM layer, which has only 256 units. Because the training ground is relatively small, there were many possible solutions potentially all interconnected forming a smooth energy landscape in the solution space. The convergence of training was therefore guaranteed in a gradient-based optimization. This follows the first part of a conventional wisdom in deep learning (Bengio, 2012; Bottou, 2012) which states in essence:

Always use a big model...

And the second part:

...with a strong regularization

If the purpose of a big model is to make optimal solution dense—hence a smooth energy landscape, a strong regularization is to steer the solution toward certain desirable region during gradient descent. However, one has to keep in mind that, sometimes, if a regularizer is too strong, it could practically change the task. For example, a training loss may saturates at a higher value, such that the final solution does not

belong to any solution classes without regularizer. In that case, whatever function hypothesis this training scheme aims towards must include a description of the regularizer too. In later discussion, I will give an example of tentative function hypothesis that includes description of regularizer.

Autoencoder or path-integrator?

Although the use of a dropout layer in Deepmind's architecture does not seem to be the most common choice[4], It did nevertheless effectively implement an information bottleneck, in the sense of (Tishby et al., 2000), similar to that of an autoencoder, in which the middle (hidden) layer[5] has much less cells than that of input and output layers. While it's known that such an autoencoder type of architecture can produce periodic tuning curves (Dordek et al., 2016; Kropff and Treves, 2008). That said, Deepmind's architecture is not entirely equivalent to this type, nor an autoencoder type of mechanism without dropout is responsible for the emergence of a modular code.

In Deepmind's scheme, they had a place cell input only being active initially. For the rest of a training epoch lasting for 2 seconds, the LSTM layer needs to path integrate. This makes an effective 30cm inter-landmark distance[6] as indicated as gray dots in Figure 2.4(a). While path-integration for 2 seconds doesn't seem to be very long, it is nevertheless significant. Taken together, it is likely that the modular periodic representation Deepmind observed in their training scheme emerged out of an interplay between the need for an effective feedforward transformation of place cell inputs and path-integration. However, the precise mechanism for such an emergence is still unknown.

Why a successful training?

To understand why Deepmind's training were successful and achieved accurate self-localization. I made a table for comparison to my earlier scheme.

In general, deepmind's scheme had a smaller environment and a powerful LSTM layer that provided twice more degrees of freedom than a simple RNN layer for more

[4]L1 or L2-norm on synaptic connectivity, or dropout directly in main computational layers are more common and straightforward forms of regularization.

[5]Although the dropout layer has 512 cells, the effective number of cells is much smaller because 50% of the cells are randomly turned off.

[6]The effective inter-landmark distance = 2s*15cm/s =30cm.

	Deepmind	Scheme 2
Spatial dimension	2D	1D
Inter-landmark distance	100*.02s*15cm/s = 30cm	50m/50 = 1m
Type of recurrent layer	LSTM	RNN
Number of cells: N	256 LSTM units	100 RNN units
Effective coding range: ℓ	215	200
$N > \ell$?	Yes	No
Noise in cells	50% dropout layer	No, or small indep. noise to each cells

Table 2.1: Deepmind's scheme vs Scheme 2. The difference was threefold: 1) smaller vs large environment, 2) LSTM vs simple RNN, and 3) dropout vs no noise-type regularizer.

flexible dynamics and easier learning. Another factor that might also attribute to an easier learning is the use of 2D environment. In 2D, every location has more neighbors (e.g. left, right, front, and back) than that in 1D. The relative abundance of neighbors together with the fact that neighbor codewords overlap, a 2D agent effectively revisited one location more frequently than a 1D agent, resulting in a faster convergence[7]. All three effects together made an overall smoother energy landscape such that a good solution is more accessible by the process of gradient descent.

The emergence was not from a task based on a function hypothesis

The training recipe provided by Deepmind was not backed by a rigorous function hypothesis. Based on their results, in hindsight, one may formulate a corresponding function hypothesis that states:

> *Grid cells exist for an accurate self-location estimate in a small environment under presence of extreme noise in circuit.*

This hypothesis doesn't sound all that plausible and is unlikely a function performed by the brain. It is also not the function hypothesis Deepmind had in mind originally (They emphasized that grid cells perform efficient vector navigation which will be discussed next.) The training scheme of Deepmind's, in the end, can neither be accounted for a search for emergence let alone an optimization principle.

Having said that, their observation is nonetheless very interesting and more detailed studies should be done in the future for understanding how such a grid cell like

[7]Revisiting a location is crucial for gradient-based optimization to incrementally reduce the error. In BPTT, this is particularly important. A relevant weight change for building up a stable codeword at one location needs to accumulate before the precious gradient signal get lost in a random walk.

code is capable of resisting a strong dropout noise.

A function hypothesis about vector navigation is not responsible for the emergence

The function hypothesis Deepmind had in mind is vector navigation. Although they did not seek for emergence based on this function hypothesis. Utilizing grid cells for vector navigation is nonetheless very interesting and totally worth investigating in future studies. In the second part of the paper, they demonstrated how a grid cell code may be advantageous for vector navigation. I would like to highlight one of their findings and make a remark regarding how it works. The actual mechanism that underlies their result might be very different from that of goal-vector decoding (a mechanism Deepmind used to explain their results).

Attempted to observe goal-vector decoding by feeding policy LSTM a codeword from goal location

In their RL scheme, Deepmind fixed the weight of a grid cell agent and only trained an external policy LSTM. The policy LSTM has access to the grid cell codeword (activity pattern) of the current location as well as that of the goal location. They claimed that the agent is inherently able to use these two codewords to decode a goal-vector (Bush et al., 2015; Stemmler et al., 2015) at any given time for a direct marching toward the goal. The evidence they provided is shown in Figure 2.5 where the agent had been trained when only the 5th door was opened. In the test runs, however, all 5 doors were opened. One can clearly see the change in the agent's policy.

No explicit goal-vector decoder and no linear-look-ahead circuitry

The claim of observing vector navigation is however questionable. First, for a real-time explicit goal-vector decoding to be possible, there must be an elaborate external decoder built in the system as stressed in Bush et al. (2015) for large-scale navigation with a combinatorial grid cell code or in Stemmler et al. (2015) for small-scale high-precision navigation with a nested grid cell code. However such a decoder is absent (or at least too implicit to identify) in Deepmind's policy network. Moreover, a real-time decoding as such in the both cases can be done with place cells alone (only with a smaller coding range or lower precision). There is no reason to expect place cell agent

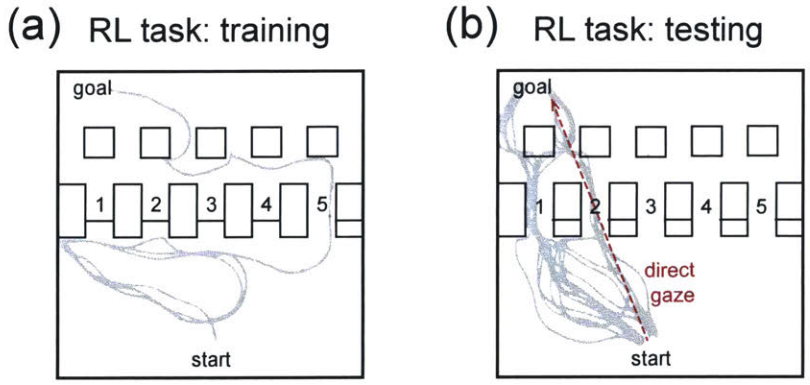

Figure 2.5: Grid cell agent perform vector navigation in Deepmind's RL scheme. **(a)** Only the 5th door was opened during training, grid cell agent has to take a detour towards goal-location. (b) The agent is capable of switching policy when a gaze toward goal-location made available after all the doors were opened in test runs. Figure re-plotted from (Banino et al., 2018)

has qualitatively different behavior (as stressed in their paper) if explicit goal-vector decoding is indeed the mechanism the policy network implemented.

The other decoding strategy requiring no elaborate decoder is called linear-look-ahead decoding (Erdem and Hasselmo, 2014, 2012; Kubie and Fenton, 2012). The decoding method uses existing path-integrator to rapidly sweep a series of mental simulation radiating outward from the current location. If one of the simulated trajectories hits the target, i.e. matches goal representation, one terminates sweeping simulation and extract goal-direction—as direction of the matching trajectory, and goal-distance—in proportion to the duration of the target trajectory.

The issue regarding using linear-look-ahead decoding is the same. The explicit circuitry wasn't there in Deepmind's policy network. Moreover, such a strategy requires extensive time for planning, which doesn't seem to happen in their test runs. The grid cell agent seem to respond immediately and marched toward goal location from time zero.

Not a goal-vector decoding but a switchable multi-policy successor representation instead

To explain their finding, I propose that the agent managed to learn more than one successor representations (SRs) (Dayan, 1993) implicitly stored in the policy LSTM over the course of training. An immediate policy switch, in Deepmind's RL scheme, after a change in visual scene (doors opened) is one evidence for this explanation.

From the beginning of training, the policy LSTM is fed with activities from a set of spatially extended tuning curves of grid cells'. If such tuning curves are thought of as spatially extended basis functions used for building an implicit set of SRs, the initial SRs and action-value function were both be extended in space and far from random (due to long range correlation of the initial SRs).

Over time, the agent started to reach the goal and gradually refine its SRs for a faster path. Those inefficient SRs, however, were not lost after training, but only become dormant in the policy LSTM. Meanwhile, the policy LSTM learnt to better associate the goal-visual input with the goal-location codeword—in such a way that this particular visual input could modify dynamics of the policy LSTM drastically—as goal-reaching events accumulated. It is therefore possible for the existence of such a goal-directed behavioral switch in the test runs without ever using explicit goal-vector decoding.

Recent studies have shown how such multi-policy SRs can be learnt in a deep neural network, and how this strategy can even be optimal in many settings (Barreto et al., 2016; Kulkarni et al., 2016).

Vector navigation as a potential functional constraint for an optimization principle

Based on this observation, it is possible to formulate a relevant function hypothesis:

> *Grid cells exist in order to build multiple successor representations for a flexible policy switching during a goal-directed navigation.*

This function hypothesis is an elaborate one. To further formulate a corresponding optimization problem requires many more judicious thoughts. The attempt will likely, however, bring a much deeper insight for vector navigation and the functions of grid cell code.

Lastly, I would like to stress that the above explanation I provided for Deepmind's finding is unlikely the full story. More study is needed and is worth to be done for a deep understanding of how all of it really works.

2.2.4 Function complementarity revisited

In this section, I would like to make a few conjectures regarding the existence of the grid-place function complementarity (introduced in Chapter 1) in the brain, based on the results of the RNN training schemes we have discussed. The conjectures goes as follows

1. *A grid cell circuit is evolutionarily hard-wired through a non-Hebbian-type plasticity mechanism during development.*

2. *A grid cell circuit—to a large degree—cannot be modified through experience via synaptic plasticity without compromising its essential function—high-capacity encoding.*

3. *A place cell circuit can only learn from experience via synaptic plasticity—with a compromise in its coding capacity.*

To see how these conjectures are plausible, ironically, we need to rely on the negative results on the attempts of searching for emergence. Specifically, The unsuccessful demonstrations of the emergence—from both mine and Deepmind's[8] RNN training schemes—may provide insights to an important question—why there exist place-grid cell duo system instead of a mono system.

Grid cell circuit solution is inaccessible via synaptic plasticity

The failure of converging to a static circuit implies rarity or inaccessibility of these solutions capable of solving the task. If we equate a gradient descent update to a biological synaptic learning[9], a grid cell circuit in the brain could as well be an island of solutions scattered sparsely among all other non-grid cell solutions, isolated, inaccessible by synaptic modification based on the animal's experience. And this ultimately explains Conjecture 2, i.e. the incapability of modifying a grid cell circuit across experiences because the other potential grid cell solutions are not reachable via synaptic plasticity rules.

[8]Deepmind's results only demonstrated emergence in a small environment.

[9]While it's debatable to draw a parallel between gradient descent connectivity update and synaptic plasticity in biological circuit, it is also not too far-fetched—from an algorithmic viewpoint—given that the two processes both search a solution though a series of incremental moves in the solution space.

One might argue that there is no need for a grid cell circuit to be modifiable; however, if one considers the fact that a place cell circuit exist in the brain and is constantly under reconstruction based on experience, it is undeniable that such an experience-dependent learnable circuit is very much needed.

Evolution might wire up grid cell circuitry without relying on synaptic plasticity

The above argument begs a question—how does a grid cell circuit come about if these circuit solution is inaccessible by synaptic plasticity? Fortunately, there is more than one way for the brain to wire up a circuit. Because a grid cell circuit—to a large degree—is static, it might as well be developed (connectivity blueprint programmed in genes) rather than be wired up via Hebbian-type synaptic strengthening (Widloski and Fiete, 2014). In other words, evolution could have millennia to work out a creative way for constructing such circuit without ever being restricted to synaptic plasticity; hence *Conjecture 1.*

Function complementarity exists due to dynamical constraints from neural circuitry

Putting all the above reasoning together, we can see how *function complementarity* plays out in the grid-place cell duo system. A grid cell circuit solution—in order to achieve high-capacity encoding—are rare and largely inaccessible via synaptic learning rules. Consequently, it has a rigid connectivity that does not change based on experience. In a drastic contrast, a place cell circuit solution—in order to achieve a learnable encoding—needs to be abundant and accessible by synaptic learning rules. In exchange for this essential function, place cells compromise its coding capacity—as we will discuss in Chapter 3. And this explains *Conjecture 3.*

2.2.5 Coding theoretic approach: Grid cells emerge for specific coding properties

Starting from the next section, we will depart from RNN training approach until Chapter 3. The focus will be on an entirely new paradigm for pursuing an optimization principle—a coding theoretic approach. As the name suggests, this approach

focuses directly on a code—a set of tuning curves—without concerning which dynamical process that generates it. This approach also takes the liberty of computing any coding related quantities that might not be shannon-type of information (e.g. Fisher information or mutual information)—hence the separation from information theoretic approach. Because of this freedom, handling a large population code could be more feasible. And by carefully choosing a coding quantity to compute, the results could also be more interpretable—from a geometric viewpoint—than that from an information theoretic approach.

In a coding theoretic approach, the tuning curves themselves are the optimization parameters instead of recurrent weights—that give rise to tuning curves—as it would be in an RNN training approach. By leaving the RNN dynamics behind, a functional constraint can be precisely specified without being tangled with dynamical constraints. Moreover, leaving dynamics behind is rather necessary if we would like to proceed using gradient-based optimization method for the reason—discussed in the last section—that a grid cell circuit solution could be inaccessible via an RNN training approach. And a coding theoretic approach, on the other hand, provides us a bypass to avoid a highly restricted solution space caused by a gradient-based learning dynamics.

That said, before diving into coding theoretic approach, one has to address an immediate issue that is seemingly in conflict with the requirement in an optimization principle—to fulfill core constraints, or to *demonstrate the existence of a neural implementation.* Luckily, for a grid cell system, the core constraint is fulfilled so long as a static circuit exists to generate desirable tuning curves. It is unnecessary to show how this circuit is wired up due to grid cells' rigid—up to a shift or rotation in grid—tuning properties across environments[10] as mentioned in Chapter 1. Further, fulfillment of the core constraint is guaranteed because the special wiring of RNNs capable of generating grid cell tuning curves have been shown (Burak and Fiete, 2009). These studies demonstrated how a stable continuous attractor can be embedded in a subnetwork, and how a full network can virtually generate any codes with arbitrarily assigned number of modules, periodicity, etc. In other words, for every set of plausible grid cell tuning curves, one can find a corresponding RNN circuit to generate it.

With the issue being addressed, a coding theoretic approach is a fully legitimate approach for pursuing an optimization principle.

[10]A place cell system in contrast has changing tuning curves. It can acquire a new set of tuning curves in a novel environment. Core constraint for place cells therefore also require demonstration of RNN sequential learnability as discussed in Chapter 3.

2.3 Which coding properties lead to grid cells?

Using an approach of RNN training to formulate and test a function hypothesis seems straightforward. For that, I used a self-localization task demanding an RNN agent to accurately path-integrate in a large environment. On the other hand, it is not immediately obvious how to formulate a function hypothesis using a coding theoretic approach. The challenge lies in abstracting a high-level description, like path-integration—or navigating—in large environment, into a set of coding properties.

In this section, I will first quantify two main properties of a binary grid cell code which provides a clue for formulating two precise coding properties for continuous neurons: 1) high-capacity and 2) translationally invariant (TI). I will explain how the TI constraint relates to path-integration and motivate the use for its potential benefit to other important high-level functions it might provides. In the meantime, I will point out why a binary neuron optimization scheme cannot answer the question of emergence—using generalization of the TI constraint[11] as an example—and it is rather necessary to generalize the problem to continuous neurons.

To proceed with this generalization, I will introduce a necessary method that uses basis functions for constructing continuous tuning curves. The basis function method not only enables a gradient-based optimization but also implants an important noise assumption in a code. I will then reason why—for continuous neurons—using separability type of capacity measure as a starting point in philosophy of having a minimal assumption on noise. And I will explain why choosing average Euclidean distance over mutual information as the capacity measure.

With both the generalization of capacity measure and TI constraint in place, I will end the section by pointing out why the two functional constraints are still not sufficient and what could be the third candidate constraint for the emergence of grid cell code.

[11]The TI constraint for continuous neurons does not exclude non-periodic solutions—or non-grid cell solutions—unlike its binary counterpart.

2.3.1 Grid cell code is translationally invariant (TI): Insight from binary neurons

We know—from our function hypothesis—that a grid cell code possess path-integration capability. But if path-integration is merely for a reliable spatial encoding, a set of random spatial tuning curves will do. In this section, I will show that a grid cell code has highly structured spatial tuning curves that supports a plausible path-integration.

A path-integrating code is not just spatially tuned

If a set of tuning curves is merely a collection of random spatial functions, the progression of a codeword will depend on the position. This could be problematic for the brain because the machinery for generating such inhomogeneous progression would need to be an elaborate one, i.e. being fundamentally state-dependent or even being codeword-specific. Because such a codeword-specific path-integrator is highly unlikely to be implemented in a biological system[12], we can assume that a path-integrating code is not of any arbitrary spatial tuning curves, but highly structured instead.

A generic constant-curvature spatial code is not grid cell code

So, which type of structure one should expect for constraining a path-integrating code? To start, I assumed that a codeword progresses in a constant rate—with respect to position—such that the curvature between two adjacent codewords is conserved:

$$\kappa(x) \equiv \left| \frac{d}{dx} z(x) \right|^2 = \text{const.} \tag{2.10}$$

where $z_i(x)$ is tuning curves of cell i. Under this constraint, If we further maximize the capacity (avoiding repeating codewords) for a binary spatial code, the outcome will not be a grid cell code. An example of such maximal capacity code is demonstrated in Figure 2.6(c). The code has 2 out of 8 cells being active at anytime, the number of possible codewords are

$$N_{\text{codewords}} = \binom{8}{2} = 28. \tag{2.11}$$

[12]To have a state-dependent path-integration, a circuit needs to not only generating these states but also remember all the different integration rules corresponding to each different state.

We can see in Figure 2.6(c) that it is, in fact, possible to have a code with a constant curvature whilst using up to all 28 nonrepeating codewords. Such code is therefore maximum in capacity, but it does not have periodic tuning curves like a grid cell code.

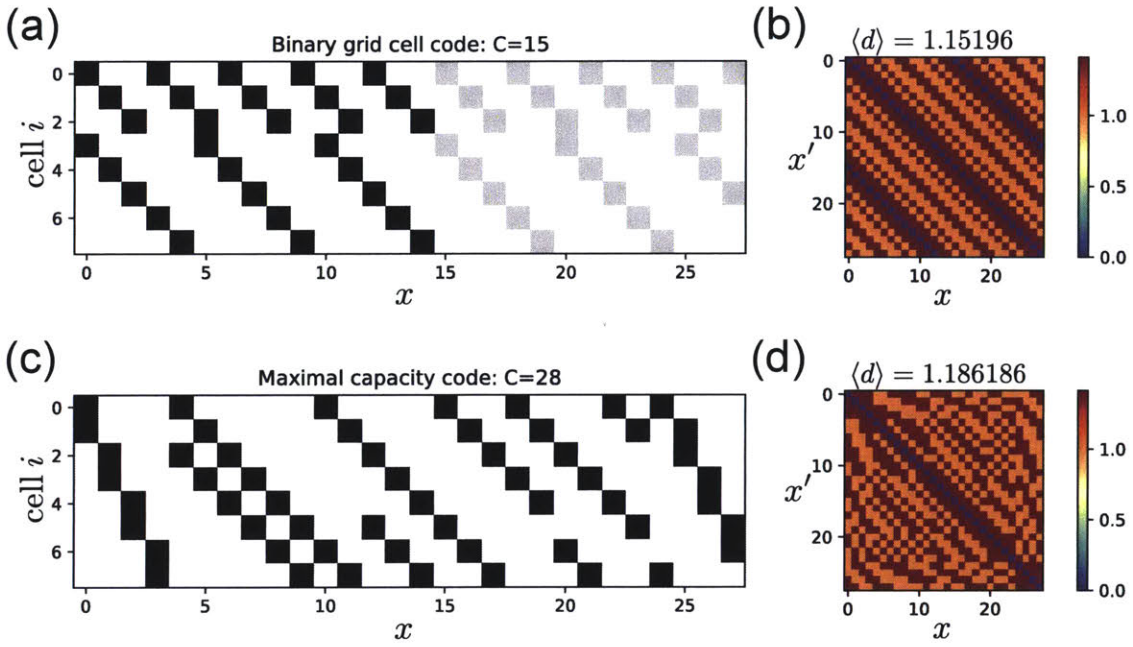

Figure 2.6: A binary grid cell code is not optimal in capacity. **(a)** A two-module binary grid cell code with coding range L=15. Gray codewords started to repeat. **(b)** Euclidean distance matrix for each pairs of codewords is constant diagonal—a manifestation of a translationally invariant (TI) code. **(c)** A maximal capacity binary code with a constant changing rate of adjacent codewords. **(d)** The corresponding distance matrix is not constant diagonal.

A binary grid cell code is TI

The implication from the above maximal capacity code is clear: a grid cell code is constrained differently from just having constant curvature. To see what this constraint might be, I plotted an example 8-cell grid cell code—partitioned into two modules: 8=3+5—for comparison. Its tuning curves are shown in Figure 2.6(a). One can see that due to the periodic organization, not all 28 possible codewords are used in the construct. And the codewords start to repeat after $x = 15$. The periodic organization of codewords indicates a globally coherent structure among not just neighboring but also distant codewords. It is therefore interesting to investigate the correlation structure for all pairs of codewords. The result is plotted in Figure 2.6(b) in which one can see a clear constant diagonal structure of Euclidean distance matrix of the code. A constant diagonal distance matrix implies the corresponding code to

be translationally invariant (TI) because $d(x, x + \Delta x) = \text{const.}$ for a fixed Δx. A binary grid cell code is therefore a TI code.

A TI code is a direct representation of modular arithmetic

It is important to stress that the binary grid cell code demonstrated above is an equivalent of modular arithmetic with a progressive constant addition. In the above example, all 15 codewords can be generated by repetitively applying a permutation transformation as follow.

$$\mathbf{z}(x) = W^x \, \mathbf{z}(0), \tag{2.12}$$

where $\mathbf{z}(0) = [1, 0, 0, 1, 0, 0, 0, 0]$ with only the first cell of each module being active, and

$$W \equiv \left(\begin{array}{ccc|ccccc} & 1 & & & & & & \\ 1 & & & & 0 & & & \\ & 1 & & & & & & \\ \hline & & & & & & & 1 \\ & & & & 1 & & & \\ & 0 & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{array}\right) \tag{2.13}$$

is a two-module permutation matrix. Each time when one apply W on codeword $z(x)$, one moves the active bumps to the next cells. The codewords generated in this process will be unique until reaching its capacity limit (the non-repeating coding range) (Fiete et al., 2008):

$$C(P_N) \equiv LCM(P_N), \tag{2.14}$$

where a certain partition $P_{N=8} = [3, 5]$ is chosen for our 8-cell example and $LCM(P_{N=8}) = 3 \times 5 = 15$ is the least common multiple of this partition $P_{N=8}$. Below I use the 8-cell example to compare the representation of modular arithmetic with the equivalent representation of binary neuron.

From this simple exercise, we can see that a binary grid cell code directly implements modular arithmetic underlies a simple path-integration machinery. We have now much more insight about what coding properties a grid cell code endows and should be ready for formulating an optimization problem to test its function hypothesis. Nevertheless, there are still issues in using binary neuron scheme for our purpose

x	x mod $(3,5)$	binary code
0	(0,0)	(100 10000)
1	(1,1)	(010 01000)
2	(2,2)	(001 00100)
3	(0,3)	(100 00010)
4	(1,4)	(010 00001)
5	(2,0)	(001 10000)
6	(0,1)	(100 01000)
7	(1,2)	(010 00100)
8	(2,3)	(001 00010)
9	(0,4)	(100 00001)
10	(1,0)	(010 10000)
11	(2,1)	(001 01000)
12	(0,2)	(100 00100)
13	(1,3)	(010 00010)
14	(2,4)	(001 00001)
15	(0,0)	(100 10000)

Table 2.2: Binary-neuron representation of modular arithmetic of a discrete position

which I briefly discuss below.

The optimization problem in binary neuron scheme is irrelevant to the emergence

In a binary neuron scheme, the TI constraint is equivalent to having modular periodic tuning curves as demonstrated in the above 8-cell code example. The remaining optimization problem is therefore only on the capacity, i.e. to maximize the coding range with no repetitive codewords. In other words, the only mission is to find the best modular periodic codes among all possible modular periodic codes. Unfortunately, this optimization problem is trivial for our purpose: seeking for emergence, because the binary neuron scheme pre-excludes all other non-modular non-periodic codes.

Issue on generalizing to continuous neurons

The above demonstration is just one reason for not dwelling on binary neuron scheme. More importantly, it is unclear whether the knowledge gained from a binary neuron scheme can be transferred to a continuous neuron scheme. For example, a TI con-

straint for continuous neurons could manifest very differently, e.g. a continuous TI code might not have modular periodic tuning curves.

Also, a continuous neuron scheme contain vastly more possible solutions from that of binary neurons. It is implausible to do an exhaustive search on all possible tuning curves. A gradient-based optimization is therefore a reasonable choice. However, for such algorithm to run efficiently, a fixed architecture is required (a changing coding range is not feasible). This means, a new capacity measure rather than counting nonrepetitive codewords needs to be considered[13].

For these reasons, we should move on and study how both high-capacity and TI coding properties can be generalized to a continuous neuron scheme.

2.3.2 TI constraint for continuous neurons

We start from generalization of the TI constraint. Recall that a binary grid cell code can be generated from repetitively applying a permutation transformation to an initial codeword.

$$\mathbf{z}(x) = W^x \mathbf{z}(0)$$

And spatial translational invariance is a direct consequence for such a code, i.e.

$$d(x, x + \Delta x) = \text{const. for a fixed } \Delta x$$

where $d(x, x')$ is the Euclidean distance between two codewords. If one adopts this type of machinery for codeword progression, then the remaining question for the generalization is straightforward: what is the matrix W that can produce a continuous TI code? The answer—as it turns out—is that W can only be an orthogonal matrix. One can see the proof I provided in the Appendix A.1, but here we focus on how such an orthogonal transformation is at work and how such a continuous code related to a binary grid cell code.

[13]For a binary neuron code, it is straightforward about what it means for two codewords to be different, i.e. $\mathbf{z}_1 \cdot \mathbf{z}_2 \neq 0$. But for continuous neurons, this distinction is blurry, and it is necessary to introduce a threshold parameter for counting differences. Consequently, this introduces a new hyperparameter into an optimization scheme which I will try to avoid in Scheme A and B, but finally adopt in Scheme C.

A TI code's manifold is generated by incremental n-rotation

In Figure 2.7, I demonstrated an example of a continuous equivalent of a binary grid cell code. We can see that the tuning curves and distance matrix in Figure 2.7(c,d) resembles those features of binary neurons in Figure 2.7(a,b). To create such a continuous code, one needs to interpolate continuously from a binary codeword to the next. It means that an incremental progression—instead of a big jump made by permutation matrix—should be implemented. Assuming it takes n incremental steps to make a jump:

$$W_{perm} = W^n, \tag{2.15}$$

where $n \gg 1$. Using the fact that W_{perm} is an orthogonal matrix, one can diagonalize it and takes n-th root on just the corresponding diagonal matrix of W_{perm} to find W. The resultant W is also an orthogonal matrix as I proved in Appendix A.1. Applying W to the initial state creates continuous tuning curves like Figure 2.7(c).

The progression made by W can be seen as an incremental rotation in N dimensional state space given that W can be block-diagonalized into a series of rotational matrices:

$$W = PBP^T, \tag{2.16}$$

where P is an orthogonal matrix. The fundamental structure of such a code is only specified by B which is composed as $N/2$ blocks of 2×2 rotational matrices regardless the viewpoint specified by P.

$$B = \begin{pmatrix} \ddots & & & \\ & \cos\theta_i & \sin\theta_i & \\ & -\sin\theta_i & \cos\theta_i & \\ & & & \ddots \end{pmatrix} \tag{2.17}$$

A code progresses with B is connected with the original though an arbitrary orthogonal projection P:

$$\mathbf{y}(x) = B^x \mathbf{y}(0), \tag{2.18}$$

where $\mathbf{y}(x) = P^T \mathbf{z}(x)$. This additional orthogonal transformation P^T do not change structure of the code because the correlation between a pair of codewords is conserved.

$$\mathbf{z}_x^T \mathbf{z}_{x'} = \mathbf{y}_x^T P P^T \mathbf{y}_{x'} = \mathbf{y}_x^T \mathbf{y}_{x'}, \tag{2.19}$$

In the coordinate frame of $\mathbf{y}(x)$, the tuning curves are simple sinusoidal waves as shown in Figure 2.7(e). Its unchanged correlation structure is plotted in Figure 2.7(f) in comparison with Figure 2.7(d).

Binary grid cell codes form a small subset among a more general set of continuous grid cell codes

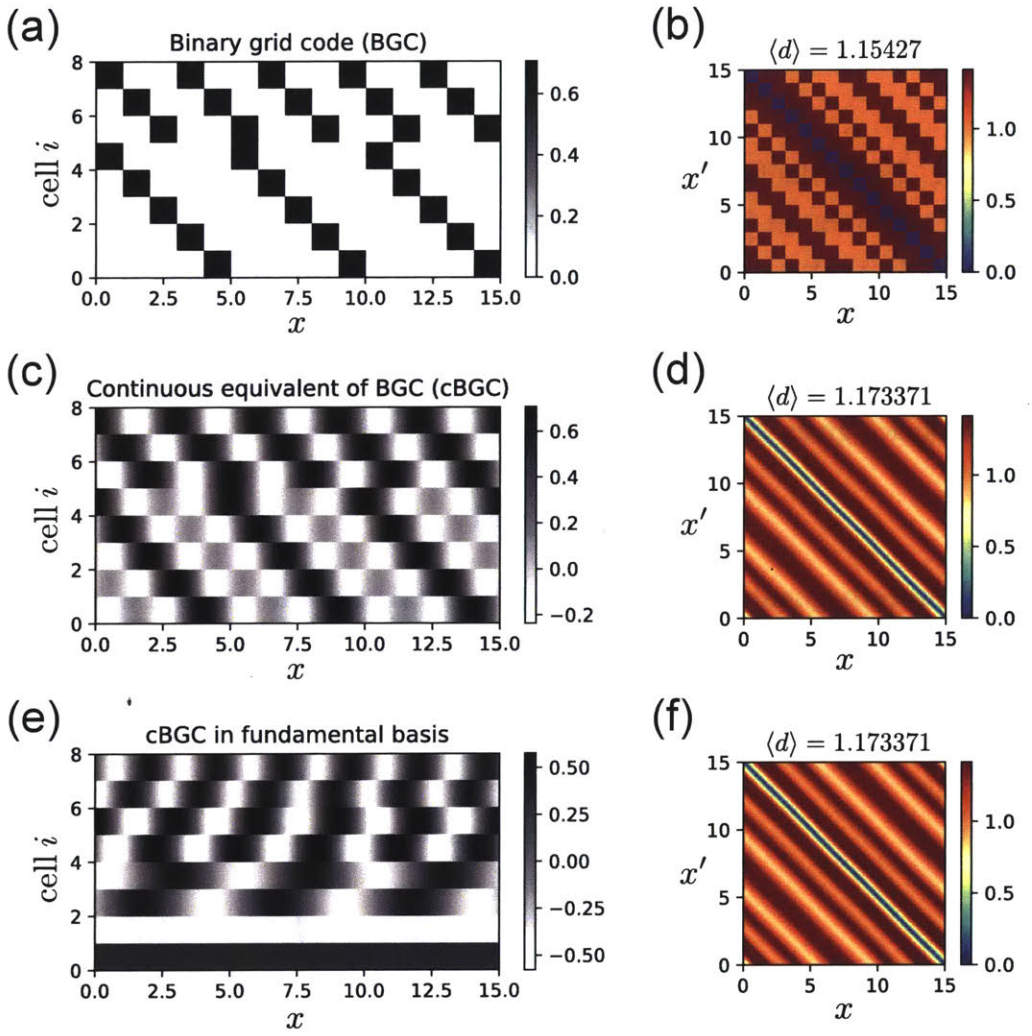

Figure 2.7: Continuous-neuron equivalent of a binary grid cell code. **(a,b)** A two-module binary grid cell code and its corresponding distance matrix. **(c)** A continuous neural code—generated via an incremental rotation of the initial codeword—that resembles the binary grid cell code in (a). **(d)** The corresponding distance matrix also displays similar sideband feature visible in (c). **(e)** Fundamentally the same code as (c) but in a different Cartesian basis. Code (e) is a result of an orthogonal transformation on code (c). **(f)** The distance matrix remain intact because (c) and (e) share the same underlying coding-line geometry in high-dimensional coding space.

From the above example, we know that all binary grid cell codes have a continuous-neuron equivalent, that is to say, all binary grid cell codes are solutions within a much larger group of continuous TI codes. An optimization problem for continuous neurons therefore encompasses all possible binary grid cell solutions as well as those potential "non-binary" grid cell solutions, e.g. a modular periodic code without frequencies being integer multiples[14].

2.3.3 Why a TI code?

Path-integration is unlikely to be codeword-specific

As we have discussed, if a code were just spatially tuned, its corresponding path-integrator has to be codeword-specific and therefore is unlikely to be implemented in a biological circuit. On the other hand, we have seen that if a code were generated via a simple path-integrator that progresses codewords using a constant orthogonal transformation, the code will naturally endows a code with TI property. However, in general, this global TI property is only true for a deterministic system. For a noisy system, it is, in fact, a labor to maintain global TI as we will discuss next.

Local TI constraint is generic

A code that is locally TI is likely to be generated merely by a generic path-integrator without much precision. As long as it progresses codewords in a constant rate, even for a noisy integrator, the short range correlation is conserved:

$$\mathbb{E}\left[\mathbf{z}_x^T \mathbf{z}_{x+\delta}\right] \approx \left\langle \mathbf{z}_x^T \mathbf{z}_{x+\delta} \right\rangle \ \forall x, \tag{2.20}$$

where $\delta \approx$minimal tuning width. A grid cell code is also locally TI. And an important function of this property is to rapidly guide the formation of a place cell code through an online similarity matching process—as we will discuss in Chapter 3 in detail. It might sound like any locally TI codes can do the job, but a grid cell code is required because of its second crucial coding property: nonrepetitive codewords over a large coding range. To see why, imagine a small segment of a grid cell code guides the

[14] All binary grid cell codeword progresses one integer step at a time. The frequencies from its continuous equivalent are therefore integer multiples for building up integer periodicity. A non-binary grid cell code can therefore be defined as a modular periodic code without frequencies being integer multiples.

formation of a map of place cells; multiple non-repeating segments of the same grid cell code therefore guide the formation of multiple maps of place cells.

Global TI constraint may be maintained for important high-level functions

A code that is globally TI requires precision in underlying circuitry. It is not difficult to see why. Imagine a noisy path-integrator that progresses a codeword with error:

$$\mathbf{z}(x+1) = (W + \xi)\ \mathbf{z}(x). \tag{2.21}$$

One then cannot expect a reliable long range correlation between codewords, i.e.

$$\mathbb{E}\left[\mathbf{z}_x^T \mathbf{z}_{x+\delta}\right] \approx \left\langle \mathbf{z}_x^T \mathbf{z}_{x+f(\Delta)\xi} \right\rangle \ \forall x, \tag{2.22}$$

where $f(\Delta)$ is an increasing function with Δ, and $\Delta \gg$minimal tuning width The long range correlation $Delta \gg 1$ naturally becomes fuzzy at the location which a codeword is supposed to encode, and the average over noisy trials will wash away the delicate sideband structure in distance matrix—e.g. Figure 2.7(d,f). The bottom line is that such elaborate global correlation structure is unlikely to happen by accident, and it might hint an important functional role.

The functional role of a globally TI code might lie in vector navigation, which is a high level function built upon this coding property. There are two decoding processes thought to be essential for vector navigation.

1. Direct goal-vector decoding (Bush et al., 2015; Stemmler et al., 2015)
2. Linear-look-ahead goal-vector decoding (Erdem and Hasselmo, 2014, 2012; Kubie and Fenton, 2012)

A code with long range correlations like grid cell code is thought to be important for both of these decoding processes. For the first decoding process, it is debatable that such long range correlations are required because—as we discussed in Section 2.2.3—a direct goal-vector decoding can also be done in a code without global correlations as long as its decoder contains the necessary spatial information. Therefore, more thought needs to be put into the first decoding process before hypothesize it to be a function of a TI code. Nonetheless, the second decoding process is uncannily a function requiring global TI constraint and can only be done in path-integrator with global precision.

Another possibility also mentioned in Section 2.2.3 is that the vector navigation may not be done by a decoding process but by building multi-policy successor representation using tuning curves of a global TI code as basis functions. But, by far, it remains understudied to determine whether such basis functions are really necessary or advantageous for a RL agent to learn a task requires vector navigation.

2.3.4 Optimization using basis functions and an inherent noise assumption

In Section 2.3.3, we see how binary neuron solutions can be incorporated into a much larger solution space: the continuous neuron solutions. However in a continuous neuron scheme, one faces an immediate challenge in formulating a computational problem: Curse of dimensionality. Because a target solution—a grid cell code—has exponential coding range with respect to the number of cells, there will be as many as $N \times \exp(\alpha N)$ number of parameters to be optimized even in just binary neuron case. Curse of dimensionality is, of course, getting worse in the case of continuous neurons. For example, if one approximates a binary bump (a square function) as a gaussian function and discretizes a standard deviation into 10 spatial steps, the optimization problem will then contain ten times more parameters—$10 \times N \times \exp(\alpha N)$—than that for binary neurons. To mitigate Curse of dimensionality, we will use an approach of basis functions to construct tuning curves throughout the rest of Chapter 2.

Use basis functions to reduce dimensionality

In a basis function approach, one constructs tuning curves as a linear sum of a fixed set of spatial basis functions: $b(x)$, that is

$$z_i(x) = \sum_j P_{ij} z_j(x), \tag{2.23}$$

where $i = 1, \ldots, N$ is cell index, and $j = 1, \ldots, N_b$ is basis function index. The dimension of the optimization problem is $N \times N_b$. To sufficiently construct an arbitrary function within a coding range: $x \in [0, L]$, one should at least have as many as $N_b \sim L = \exp(\alpha N)$ basis functions. With this choice, the dimensionality goes back to that of the binary neuron case. The reason of using this many basis functions is straightforward. It directly follows the idea of generalizing a binary neuron scheme

to a continuous one whilst preserving its coding properties. The generalization considers an upper-bound in spatial resolution $1/\sigma$ of a tuning curve. For example, Figure 2.7(b) shows a continuous equivalent of a binary grid cell code. One can easily imagine an effective way to construct such tuning curves is to use a set of identical gaussian functions with their standard deviations to be σ which is half the width of a binary neuron's square tuning bump. Therefore, to tile the entire coding range, one also needs L of such gaussian functions equally distributed in space.

Inherent noise assumption

The introduction of a fixed upper-bound in spatial resolution has an important implication. It makes a fundamental assumption about noise in a system. To understand this, consider a set of tuning curves of a system which is generated via two abstract steps: 1) a dynamical process that reliably produces spatially tuned firing rates—or simply tuning curves, and 2) a sampling process that generates spikes out of the underlying tuning curves. If we further assumed that Step 1 is deterministic and the only noise source in the system is from Step 2, then the spatial resolution of a population code is correlated with tuning width in various ways depending on the dimensionality of an encoded variable (Pouget et al., 1999; Zhang and Sejnowski, 1999). For 1D tuning curves, spatial resolution (measured by Fisher information) increases with decreasing tuning width. For 2D, spatial resolution is independent of tuning width. For 3D and above, spatial resolution increases with increasing tuning width.

The above study has seemingly concluded that there is no dependence between optimal tuning width and noise. If so, our use of basis function approach is not justifiable without introducing another noise source—because tuning width itself does not represent noise. However, if one considers that an intrinsic noise exists in Step 1—that is, the dynamical process underlies tuning curve generation is noisy (Burak and Fiete, 2012)—then the dependence of spatial resolution on tuning width changes (Yoo, 2014). Now for all dimensions, spatial resolution increases with decreasing tuning width: σ until saturation after $\sigma < \sigma_{noise}$. This result shows a near one-to-one correspondence between optimal tuning width and intrinsic noise.

One-to-one correspondence

To finally establish a one-to-one correspondence between tuning width and intrinsic noise, we need to know how the above spatial resolution is calculated (Yoo, 2014). In

the semi-analytic calculation of the spatial resolution, an assumption of enough spike statistics is made, i.e. $N_{spk} \gg 1$ where N_{spk} is number of spikes in a unit time within a tuning width. This assumption puts the result away from sampling failure; in other words, there is no consequence of using an arbitrary narrow tuning width even exceeding the saturation of spatial resolution. However, if we also consider this sampling failure in a system, decreasing tuning width—after the saturation: $\sigma < \sigma_{noise}$—will cause a drop in spatial resolution. So there we have it, a one-to-one correspondence between optimal tuning width σ and intrinsic noise σ_{noise} as a result of balancing two opposing forces: 1) to increase spatial resolution via decreasing tuning width and 2) to maintain spatial resolution via avoiding sampling failure. Below I demonstrated this correspondence numerically with an analytical upper-bound on spatial resolution limited by minimal tuning width, or—as we established just now—*intrinsic noise.*

Spatial resolution is bounded and scales with intrinsic noise

In Figure 2.8, I numerically computed the curvature (equivalent to Fisher information) of a population code as a direct measure of spatial resolution:

$$\kappa(x) \equiv \left| \frac{d}{dx} z(x) \right|^2, \tag{2.24}$$

where $z_i(x)$ is gaussian tuning curves that uniformly tile a finite range: $[-L/2, L/2]$. As one can see, at $x = 0$ (away from the coding boundaries), the curvature saturates roughly after $N \approx 10$ cells ($N \approx L/\sigma = 10/1$). The upper-bound of curvature after saturation is

$$\kappa\left(x = 0 \mid N \to \infty\right) = \frac{1}{8\sqrt{2}\sigma} \tag{2.25}$$

The equation above shows that an optimal curvature is inversely proportional to the tuning width. One can interpret this result as a tradeoff between noise robustness and spatial resolution. The larger the noise, the wider the tuning width, σ, needs to be for robustness, and thereby, the smaller the spatial resolution—κ. This relation is demonstrated in Figure 2.9 where it shows an inverse linear tradeoff given enough cells (region away from (b)) and given that tuning curves being away from coding boundaries (region away from (d)).

These results show that the spatial resolution of a code (which is closely related to the capacity of the code as we will discuss later) is fundamentally bounded by the intrinsic noise represented by minimal allowable tuning width σ.

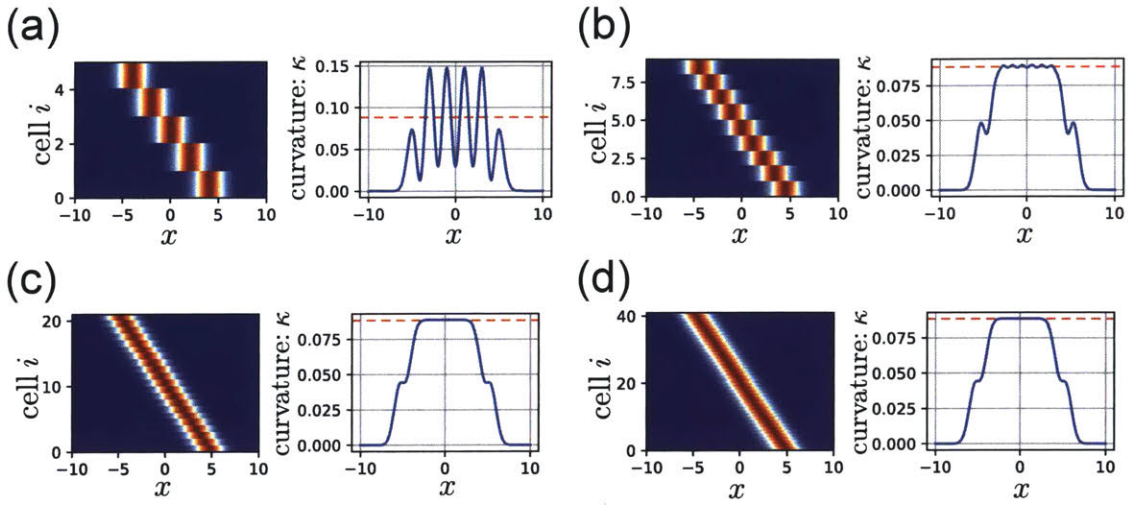

Figure 2.8: Spatial resolution saturates with increasing number of cells. **(a)** 5 cells have the same but shifted gaussian tuning curves with minimal width set by intrinsic noise level. The curvature—as a measure of spatial resolution—oscillates around analytic bound plotted as red dashed line. **(b-d)** As number of cells increases, the curvature (at $x = 0$) saturates and converges to analytic bound that is only a function of tuning width σ.

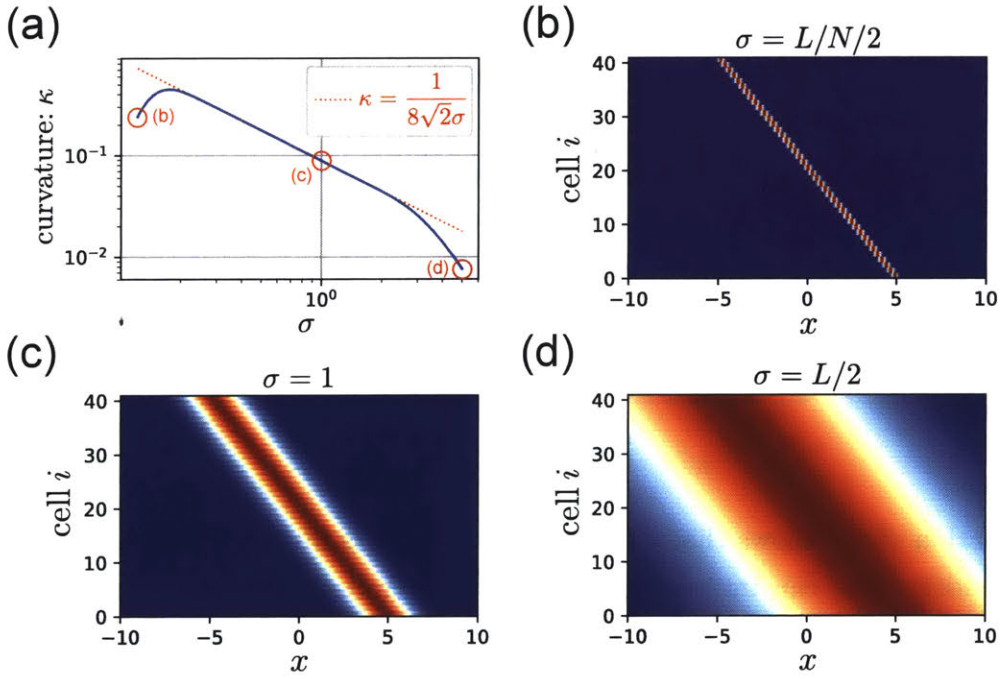

Figure 2.9: The upper-bound of spatial resolution scales as $1/\sigma$. **(a)** The curvature—as a measure of spatial resolution—matches the analytic bound: $\kappa \sim 1/\sigma$ given enough cells (region away from (b)) and without strong boundary effect (region away from (d)). **(b-d)** Three example tuning curves highlighted in (a).

Why using Fourier basis functions

So far, we successfully establish the relation between minimal allowable tuning width and intrinsic noise. From the discussion, it is natural to use gaussian basis functions

with the same standard deviation σ at any locations within the coding range. Nevertheless, we will proceed using Fourier basis functions throughout the chapter for the following two reasons:

1. *Fourier basis functions potentially provide a smoother energy landscape for gradient-based optimization to reach global optimum.* Because of the distributed nature of these periodic basis functions, a local change in codeword $z(x)$ via changing a few coefficients in P can result in changes of all other codewords $z(x')$. Therefore, a gradient-based optimization (incremental change in P) can quickly explores globally-diverse tuning curves, suitable for our interest—targeting various multi-field tuning curves.

2. *Fourier basis functions are the most natural basis functions for a TI code such that they provide a sparse coefficient matrix.* As shown in the last section, a TI code is essentially a series of incremental high-dimensional rotations on an initial codeword. If one rotates a TI code to align to its fundamental coordinate, the subsequent tuning curves are simple sine and cosine functions, and hence Fourier basis functions. If the outcome tuning curves are the basis functions themselves, the coefficient matrix P will be sparse, and thereby one may apply suitable optimization techniques exploiting the spareness of P for approaching general good solution regions.

Intrinsic noise is represented by maximal frequency

The use of Fourier basis functions does not change the intrinsic noise assumption. To see this, I maximized the sparseness of a tuning curve constructed by a set of Fourier basis functions, and showed that the inverse maximal frequency of basis functions linearly correlates with the narrowest achievable tuning width. The tuning curve is constructed via Fourier basis function as follows:

$$z_i(x) = \sum_{j=1}^{N_b} P_{ij}\ b_j(x), \tag{2.26}$$

where $i = 1, ..., N$, $x \in [0, L]$, and $b_j(x) \equiv \cos(2\pi f_{u_j} x + \phi_{v_j})$

with $\begin{cases} u_j & = 1, ..., U \\ v_j & = 1, ..., V \\ j & = 1, ..., N_b \equiv UV \end{cases}$. The frequencies and phases are equally divided: $f_{u_j} \in$

$[0, 1/w]$ and $\phi_{v_j} \in [-\pi, \pi]$. The maximal frequency $1/w$ of Fourier basis functions sets strength of intrinsic noise that limits the narrowest allowable tuning width 'sigma. The loss function for this optimization problem is

$$\text{Loss} = \frac{1}{NL} \sum_{i,x} ([z_i(x)]_+ - \alpha[z_i(x)]_-) \tag{2.27}$$

When $\alpha = 1$, the loss function is reduced to an average of $L1$-norm. Here I used $\alpha = 10$ to encourage nonnegative tuning curves. In Figure 2.10, I ran 400 random initial P to cover various w. The resultant tuning curves are fit to a gaussian function to extract the tuning width σ. It is clear that the tuning width is directly bounded by and proportional to the inverse maximal frequency w:

$$\sigma = \gamma \ w \tag{2.28}$$

The maximal frequency $1/w$ of a chosen Fourier basis functions therefore also assumes an intrinsic noise just like that of a gaussian basis function. In the later optimization scheme (Scheme B), we will use this linear relation to derive a separability upper-bound in the large N limit as a yardstick for the case of finite N.

2.3.5 Which capacity measure? Euclidean distance vs mutual information

Use a capacity measure with least hyperparameters

The capacity measure used in the earlier binary neuron scheme demands no repeating codewords. As for continuous neurons, such capacity measure requires an additional threshold parameter. In philosophy of seeking minimal sufficient conditions for emergence (discussed in Chapter 1), we should try to avoid introducing any more hyperparameters unless it's absolutely necessary[15]. In this guidance, there are two immediate candidate capacity measures that have been widely used: 1) Average Euclidean distance and 2) mutual information. In the following example, I will show how they share similar characteristics and why average Euclidean distance is potentially a better choice.

[15]Later in Scheme C, we will return to and discuss such a capacity measure with a threshold noise.

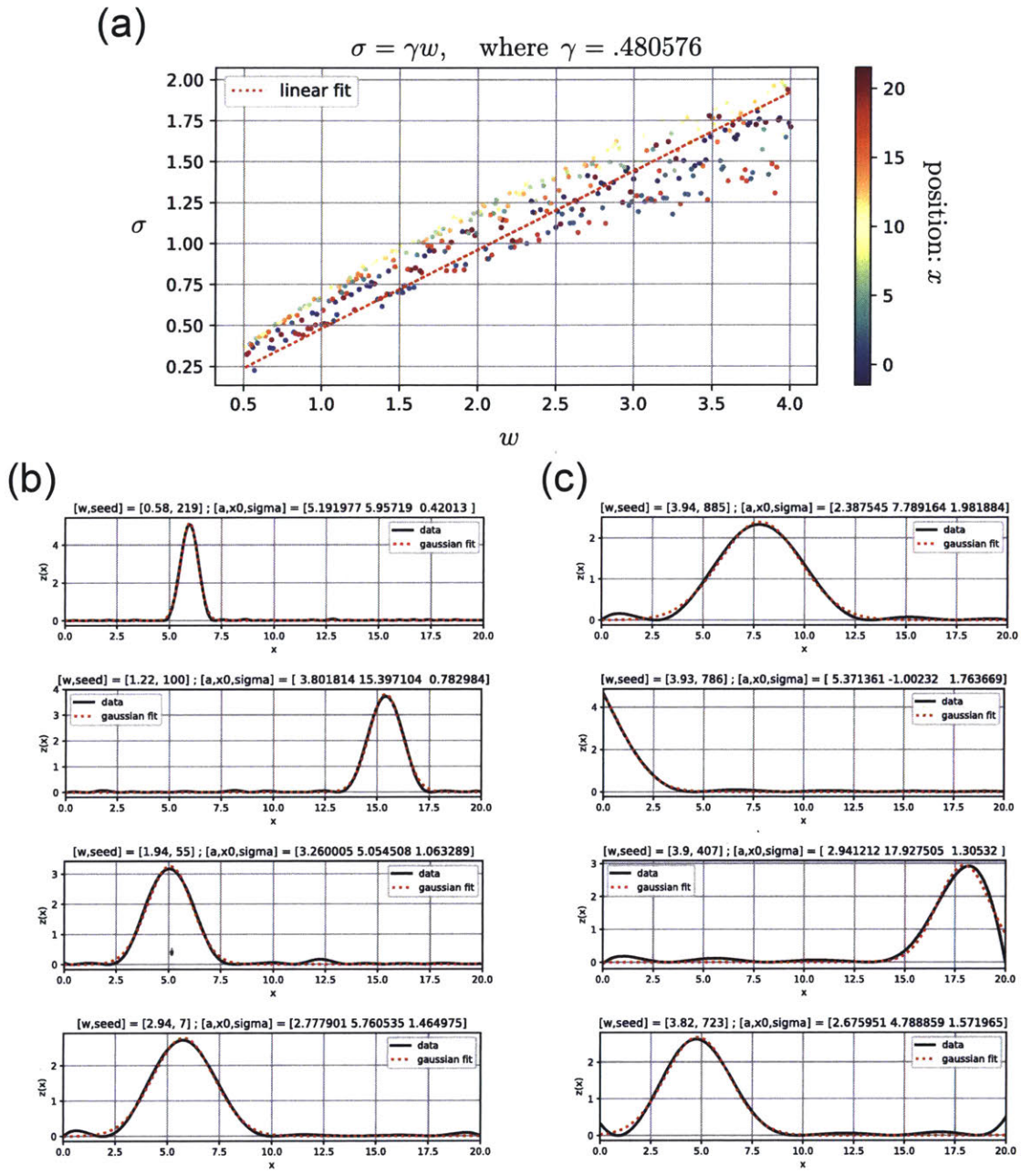

Figure 2.10: Linear relation between intrinsic noise σ and inverse maximal frequency w in Fourier basis functions. The sparsity of a single cell tuning curve is maximized via $\min_P |\mathbf{z}(x)|$—where P is Fourier coefficient matrix—under a soft-nonnegative tuning constraint. **(a)** The results of 400 optimizations—with random initial coefficient matrix and a sweep in w—are plotted. The tuning width σ is extracted from a gaussian fit. An approximate linear relation: $\sigma = .48w$ will be adopted for establishing another analytic bound for capacity in Figure 2.21. **(b)** 4 examples of σ growing with increasing w. **(c)** Four examples of the similar w but different σ due to the boundary effect.

Definitions

Euclidean distance matrix is a collection over all pair of codewords:

$$D(x,x') \equiv D(\mathbf{z}(x),\mathbf{z}(x')) = \sqrt{\sum_i |z_i(x) - z_i(x')|^2} \tag{2.29}$$

The mean of this distance matrix is used as a scalar capacity measure:

$$\langle d \rangle \equiv \frac{1}{L^2} \sum_{x,x'} D(x,x') \tag{2.30}$$

To implement a capacity measure as mutual information (MI) whilst being compatible to the use of basis functions, we need to express MI as a function of tuning curves alone. The MI between a random location X and activity pattern $\mathbf{N}$ is given as

$$I(X;\mathbf{N}) = H(X) - H(X|\mathbf{N}), \tag{2.31}$$

where

$$H(X) \equiv -\sum_x P_X(x) \log P_X(x) \tag{2.32}$$

$$H(X|\mathbf{N}) \equiv -\sum_{\mathbf{n}} P(\mathbf{n}) \sum_x P_{X|\mathbf{N}}(x|\mathbf{n}) \log P_{X|\mathbf{N}}(x|\mathbf{n}) \tag{2.33}$$

are the entropies. x is position and $\mathbf{n}$ is activity pattern generated from a poisson-spiking process that samples spikes from the underlying tuning curves $\mathbf{z}(x)$. To simplify $I(X;\mathbf{N})$, we need to first simplify a log conditional probility function $P_{X|\mathbf{N}}(x|\mathbf{n})$.

$$\begin{aligned} \log P_{X|\mathbf{N}}(x|\mathbf{n}) &\propto \log \left[\left(\prod_{i=1}^{N} z_i(x)^{n_i} \right) \exp\left(-\sum_{i=1}^{N} z_i(x) \right) \right] \\ &= \sum_{i=1}^{N} n_i \log z_i(x) - \sum_{i=1}^{N} z_i(x) \end{aligned} \tag{2.34}$$

In the above, I used the result from (Zhang et al., 1998) with assumptions of a uniform prior and a unity time window $\tau = 1$. To further reduce the computation cost, I assumed that a possible activity pattern $\mathbf{n}$ encompasses only those from the

tuning curves $\mathbf{z}(x')$:

$$\text{logP}(x, x') \equiv \log P_{X|\mathbf{Z}}(x|\mathbf{z}(x')) \propto \sum_{i=1}^{N} z_i(x') \log z_i(x) - \sum_{i=1}^{N} z_i(x) \tag{2.35}$$

We can now simplify MI from Equation (2.31), $I(X;\mathbf{Z})$ is interpreted as a measure that tells us how much—on average—an activity pattern $\mathbf{z}(x')$ conveys about a location x:

$$\begin{aligned} I(X;\mathbf{Z}) &= H(X) - H(X|\mathbf{Z}) \\ &= -\sum_{x} P_X(x) \log P_X(x) - \frac{1}{L} \sum_{x,x'} P_{X|\mathbf{Z}}(x|\mathbf{z}(x')) \log P_{X|\mathbf{Z}}(x|\mathbf{z}(x')) \\ &= \log L - \frac{1}{L} \sum_{x,x'} \exp(\text{logP}(x, x')) \cdot \text{logP}(x, x'), \end{aligned} \tag{2.36}$$

where $P(x) = 1/L$ is used and $\text{logP}(x, x')$ can be numerically computed from Equation (2.35). To compare to the distance matrix $D(x, x')$, I also define an equivalent matrix for MI:

$$M(x, x') \equiv \exp(\text{logP}(x, x')) \cdot \text{logP}(x, x') \tag{2.37}$$

Why choosing average Euclidean distance as capacity measure

In Figure 2.11, I compared MI with average Euclidean distance for two different classes of tuning curves. The top row are modular periodic tuning curves (of grid cells), the corresponding distance matrix $D(x, x')$, and MI matrix $M(x, x')$. The bottom row are for unimodal tuning curves (of place cells). From these results, we see that $D(x, x')$ and $M(x, x')$ are qualitatively the same. They both compute how separable a pair of codewords is. Maximizing either quantity, as a separability, tends to reduce overlaps among all codewords. However, in the case of modular periodic tuning curves, Euclidean distance matrix is more sensitive to small differences between codewords—resulting in more visible sidebands, whereas MI matrix only displays a clear sideband when the difference is large enough. Overall, this discussion shows that using average Euclidean distance as capacity measure has two advantages:

1. Euclidean distance is more sensitive to small tuning curve change, and it is simpler both computationally and interpretively in a geometric viewpoint.

2. Unlike MI, Euclidean distance is well defined for both positive and negative

tuning curves[16]. Using a hard constraint on nonnegative tuning has the following two issues: a) It makes gradient-based optimization more difficult[17]. And b) TI condition is hard to be met because a typical TI code is not strictly positive (Appendix A.2). In other words, many high-capacity TI code that is mostly—but not a-hundred-percent—nonnegative will be excluded by the optimization scheme.

For the above reasons, I used the average Euclidean distance as capacity measure for the following optimization schemes. At this point, we have two functional constraints finalized, i.e. 1) maximizing capacity as average Euclidean distance and 2) minimizing the deviation from being a TI code.

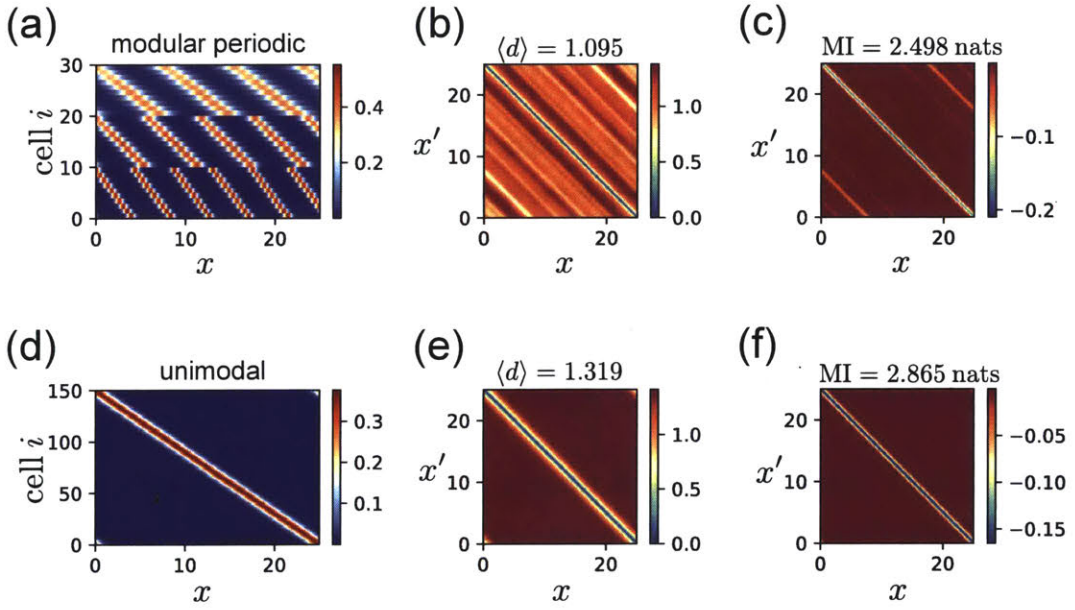

Figure 2.11: Capacity measure as average Euclidean distance or mutual information. **(a-c)** From left to right: a modular periodic tuning curves, its corresponding Euclidean distance matrix $D(x, x')$, and its mutual information matrix $M(x, x')$. **(d-f)** Same measures but for a unimodal tuning curves.

[16]In computing MI, one needs to compute log-probability function, which has to be strictly positive. And since firing rates—hence tuning curves—are directly interpreted as probability density, they have to be strictly positive too.

[17]Restricting search region of a solution to only the first quadrant in state space creates rough energy landscape, because the learning trajectory cannot bypass those restricted state space which may potentially provide a low-energy path towards global optimum.

2.3.6 A biological constraint is needed

To end this section, I summarized what we have so far and what is still missing for a full optimization problem.

What we have

1. We have learnt that an optimization problem in binary neuron scheme cannot address the question of emergence, so that an optimization scheme for continuous neurons needs to be established.

2. We have established two functional constraints for continuous neurons via a) defining a capacity measure as average Euclidean distance and 2) generalizing the binary TI constraint by associating it to incremental high-dimensional rotations on codewords.

What is missing From knowing these, one can construct two corresponding loss functions: Loss 1 that maximizes the capacity and Loss 2 that minimize the variance along diagonals of distance matrix[18]. The issue however is that the two functional constraints are not yet sufficient for the emergence of grid cell code.

From Section 2.3.2 and Appendix A.1:, we know that a TI code under an arbitrary rotation is still a TI code. However the tuning curves can be completely off after this rotation. For example, if a modular periodic code $\mathbf{z}(x)$ appears after optimization, an extra rotation on the entire code via a random orthogonal matrix O will mess up the clean periodicity of tuning curves: $\mathbf{y}(x) = O\mathbf{z}(x)$[19], whereas preserving the correlation structure across codewords:

$$\mathbf{z}^T\mathbf{z} = \mathbf{y}^T O^T O\mathbf{y} = \mathbf{y}^T\mathbf{y} \tag{2.38}$$

That is, both Loss 1 and Loss 2 are unchanged after O, but the tuning curves are no longer of grid cells.

To conclude, a third loss function is necessary for the emergence of grid cells. We hypothesize that the third loss function specifies a biological constraint and explore various forms of Loss 3 that may lead to the emergence of grid cell code. Specifically, in Scheme A, I used Loss 3 to encourage a dense code that eventually give rise to a

[18]More details in the next section.

[19]Tuning curve for a cell in $\mathbf{y}(x)$ is random linear sum of all tuning curves in $\mathbf{z}(x)$. If there were multiple frequencies in $\mathbf{z}(x)$, Tuning curves in $\mathbf{y}(x)$ will not be periodic (or single frequency).

set of sinusoidal tuning curves, and in Scheme B and C, I used Loss 3 to encourage a sparse code targeting a von Mises type of tuning curves (locally narrow and gaussian-like, but globally periodic).

2.4 Scheme A: Dense code with a linear denoising projection

Early on in Section 2.3.2 (and Appendix A.1), we know that every TI code can be uniquely (up to a constant phase freedom) represented via a set of sinusoidal tuning curves. In Scheme A, I used this fact to motivate such simple periodic tuning curves as targets. I will show why a denoising model is needed on top of three loss functions for module formation. We will begin from an ad hoc scheme, Scheme A-1, meant to demonstrate what advantages a coding theoretic approach can offer, and end with Scheme A-2 which successfully eliminates the issues present in Scheme A-1.

2.4.1 A candidate biological constraint targeting sinusoidal tuning curves

From the observation that a TI code can be uniquely represented as a set of $N/2$ pairs of sinusoidal tuning curves (Appendix A.1), one may hypothesize that this particular representation among all the equivalents (up to a random orthogonal transformation) has lowest energy in the grid cell optimization problem.

$$z_{2i}(x) = \cos(2\pi f_i x + \phi_i) \tag{2.39}$$

$$z_{2i+1}(x) = \sin(2\pi f_i x + \phi_i) \tag{2.40}$$

It's then interesting to ask which type of biological constraint Loss 3 should implement such that this unique representation is optimal. Below I provide one candidate Loss 3 that is used in both Scheme A-1 and A-2.

$$\text{Loss 3} = \frac{1}{L^2} \sum_{i,x} z_i^4(x) \tag{2.41}$$

This loss function puts a strong penalty on a few particularly large activities in $\mathbf{z}(x)$, it thereby promotes a dense code that doesn't have any large outliers in activity. One

can interpret Loss 3 as sparseness of the tuning curves to be minimized. The reason I used power of 4 instead 2 is because a generalized firing rate[20]

$$r_i(x) \equiv z_i^2(x) \tag{2.42}$$

is used and constrained such that

$$\sum_{i,x} r_i(x) = L. \tag{2.43}$$

For this reason, one needs to go to next power for defining sparseness. To see how minimizing Loss 3 can lead to simple sinusoidal tuning curves, imagine summing two simple sinusoidal functions with different frequencies. Any possible linear sums of these two functions move the code away from the target and result in two more complicated functions due to constructive or destructive interference of waves. As a result, Loss 3 goes up.

As for a biological motivation of using Loss 3, one could interpret such constraint as to avoid costly high activity. But here we will simply explore and try to understand how such constraint gives rise to the target tuning curves. The biological plausibility of Scheme A will be discussed in the end of this section.

2.4.2 A mechanism for module formation is needed

By far we have all three loss functions in place such that an optimal code is expected to be periodic tuning curves. However, to declare the emergence of grid cell code, the code has to be modular—clustering in tuning frequencies—too. In this section, I will demonstrate that having these three loss functions alone is not sufficient to give rise to modularity. An additional mechanism for module formation is required.

We have known that Loss 3 is to "rotate" a code to its most dense representation without changing Loss 1 and 2. Loss 3 alone therefore does not change the fundamental codeword structure; hence no preference towards modularity. Meanwhile, neither does Loss 2 attribute to creation of modules since it only demands a TI code in which each tuning curve has the liberty of having an arbitrary frequency. What remains less obvious is the contribution of Loss 1. To gain more insight about how Loss 1 may (or may not) lead to module formation, I mapped out the entire "energy landscape"

[20]The notion of firing rate here is related to the length of a codeword. The signal itself cannot be interpret as firing rate because we don't restrict tuning curves to be strictly positive.

of Loss 1 for all possible frequency combinations for 4- and 6-cell TI codes.

Recall that our capacity measure is defined as average Euclidean distance:

$$\langle d \rangle \equiv \frac{1}{L^2} \sum_{x,x'} \sqrt{|\mathbf{z}(x) - \mathbf{z}(x')|^2}, \tag{2.44}$$

where the tuning curves

$$\mathbf{z}(x) \equiv \begin{bmatrix} \vdots \\ z_{2i}(x) \\ z_{2i+1}(x) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \cos(2\pi f_i x + \phi_i) \\ \sin(2\pi f_i x + \phi_i) \\ \vdots \end{bmatrix} \tag{2.45}$$

are simple sine and cosine functions such that Loss 2 vanishes ($\mathbf{z}(x)$ is a TI code by design). To effectively maximize the the capacity, I used logarithmic of geometric mean in Loss 1:

$$\text{Loss 1} \equiv -\frac{1}{L^2} \sum_{x,x'} \log \sqrt{|\mathbf{z}(x) - \mathbf{z}(x')|^2} \tag{2.46}$$

2.12(a) shows a map of Loss 1 for a 4-cell TI code, i.e. two frequencies. The first pair of cells has unity frequency: $f_1 = 1$, and the frequency of the second pair is plotted as horizontal axis. The different colors label the case with different coding range L—measured by the minimal wavelength of basis functions: $1/f_1$. We can see that as $L \gg 1$, 1) the boundary effect become negligible for most of the frequencies, and 2) locally good solutions (lower value in Loss 1) form flat valleys with their Loss 1 approaching to global optimum (all locally flat valleys share approximately the same value), and 3) the bad solutions (peaks) satisfy the condition that two frequencies are multiple integers of each other, or their ratio is a rational number such that the corresponding code possess strong repetition.

$$f_2/f_1 \in \mathbb{Q} \tag{2.47}$$

Similarly for the 6 cell cases plotted in 2.12(b,c), the bad solutions (red spots) are of rational frequency ratio. The reason is simple, if the frequency ratio are rational, after coding range exceeds the least common multiple of their corresponding wavelength, the codewords started to repeat, and hence lower in capacity.

These results put us in a difficult position, because it implies that a modular code—having so many cells reuse just few frequencies—is a bad solution, and hence

a local maximum in Loss 1. If we follow the current scheme and proceed the optimization for a case with more cells, say $N = 100$, the optimal solution will have as many as $N/2 = 50$ different frequencies, which is in a drastic contrast to real grid cells—often have only 3 to 10 modules in a rodent's brain. Next, we will use an ad hoc optimization scheme to demonstrate how modular solutions can be advantageous if a denoising mechanism is introduced.

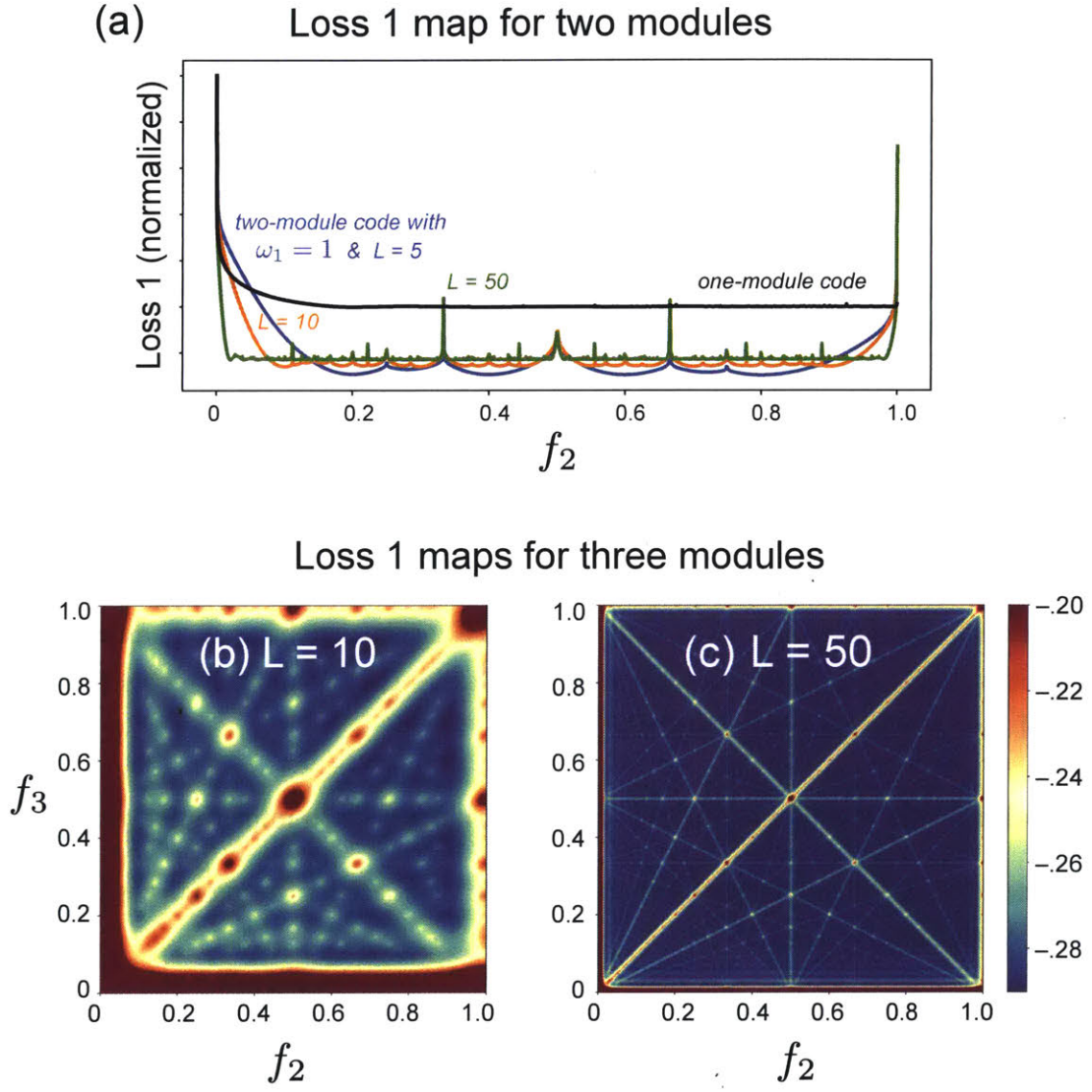

Figure 2.12: A maps of energy—inverse capacity as Loss 1—landscape for 4- or 6-cell TI codes. Good solutions cover most of the landscape. **(a)** Landscape of a 4-cell TI code with fixed frequency for first cell pair $f1 = 1$. **(b,c)** Landscape of a 6-cell TI code with coding range $L = 10wand50w$ respectively.

2.4.3 Ad hoc denoising scheme: Advantages of coding theoretic approach

In the current scheme, having a modular code is equivalent to having redundancy in the code. That is, multiple pairs of tuning curves share one frequency. To make such redundancy advantageous, a denoising mechanism is necessary. The most simple denoising model is to linearly sum many similar—yet noisy—signals such that the signal-to-noise ratio (SNR) of the output signal is much larger than the original input signals. To see how it works, consider two random numbers, $X \sim \mathcal{G}(\mu_X, \sigma_X^2)$ and $Y \sim \mathcal{G}(\mu_Y, \sigma_Y^2)$, representing two noisy signals. The sum of them is also a random variable:

$$Z = X + Y \sim \mathcal{G}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2) \tag{2.48}$$

with both the mean and variance sum up linearly. Note that, however, SNR is doubled if the two signals were the same, i.e.

$$Z = 2X \sim \mathcal{G}\left(2\mu_X, 2\sigma_X^2\right) \tag{2.49}$$

$$\implies SNR(Z) = \frac{(2\mu_X)^2}{2\sigma_X^2} = 2\ SNR(Z) \tag{2.50}$$

For N neurons case, one can consider a similar linear denoiser that inclines to sum up similar tuning curves for increasing SNR. With this denoising mechanism, one can expect a tradeoff between the following two forces that ultimately gives rise to a couple of modules in an optimal solution:

1. Each module within a code tends to recruit more cells to improve its own SNR.
2. A code, as a whole, prefers more modules via diversify its tuning curves to avoid repetitive codewords.

And precisely how many modules are needed should depend on the noise level. Specifically, a qualitatively correct tradeoff should satisfy the following two conditions:

1. Number of modules should decrease with increasing noise level.
2. Overall capacity should decrease with noise level.

Following these guidelines, the first denoising model, Scheme A-1, uses an ad hoc correlated noise term—in addition to the uncorrelated noise—as a turning knob for

controlling the allowable denoising capability. This ad hoc approach serves as a stepping stone for searching a minimal denoising model that finally implements the tradeoff correctly as we will discuss in Scheme A-2.

In philosophy of finding a minimal model, we may tentatively use the coefficient matrix—that generates tuning curves—itself as a denoiser. In so doing, both the uncorrelated and correlated noise is added at the level of Fourier basis functions[21]. Below is the setup of the optimization problem step by step.

Setup of Scheme A-1

I. Building tuning curves using Fourier basis functions. The tuning curve of i-th cell is

$$z_i(x) = \sum_{j=1}^{N_b} P_{ij}\ a_j\ b_j(x), \tag{2.51}$$

where $i = 1, ..., N$, $x \in [0, L]$, and $b_j(x) \equiv \cos(2\pi f_{u_j} x + \phi_{v_j})$ with $\begin{cases} u_j & = 1, ..., U \\ v_j & = 1, ..., V \\ j & = 1, ..., N_b \equiv UV \end{cases}$. The frequencies and phases are equally divided: $f_{u_j} \in [0, 1/w]$ and $\phi_{v_j} \in [-\pi, \pi]$.

II. Denoising model

$$\sigma_i^2(x|\beta) = c^2 \sum_j P_{ij}^2\ 1_j(x) + \beta/\sqrt{N}, \tag{2.52}$$

where $c^2 \equiv \frac{1}{L}\sum_{j,x}\ a_j^2\ b_j^2(x) \approx 1$ is a normalization factor, and $\mathbf{1} \equiv \{1_j(x)\}$ is a matrix with all entries equal 1. The matrix $\mathbf{1}$ simulates an unbiased noise strength for all cells at any locations. β controls the level of the ad hoc correlated noise, which will be used to explore the tradeoff discussed earlier.

III. Separability as capacity measure. To explore the effect of noise, I used a noise-normalized codewords instead of that from the original tuning curves.

$$\tilde{z}_i(x|\beta) = \frac{z_i(x)}{\sigma_i(x|\beta)} \tag{2.53}$$

[21]We will see in the next section how to implement the tradeoff using a more plausible denoiser with noise being added at the level of tuning curves.

And the separability is defined as the average Euclidean distance between all pairs of these renormalized codewords:

$$D\left(\tilde{\mathbf{z}}(x|\beta), \tilde{\mathbf{z}}(x'|\beta)\right) = \sqrt{\sum_i \left(\tilde{z}_i(x|\beta) - \tilde{z}_i(x'|\beta)\right)^2} \tag{2.54}$$

IV. Three loss functions. The first loss function maximizes separability:

$$L1(\beta) = -\frac{1}{L^2} \sum_{x,x'} \log D\left(\tilde{\mathbf{z}}(x|\beta), \tilde{\mathbf{z}}(x'|\beta)\right) \tag{2.55}$$

The second loss function is a soft version of constraining the code to be translationally invariant (TI):

$$L2 = \frac{1}{L} \sum_{\Delta=0}^{N-1} \mathrm{Var}(\mathbf{D}(\Delta)), \tag{2.56}$$

where $\mathbf{D}(\Delta) \equiv \{D(\mathbf{z}(x), \mathbf{z}(x+\Delta))\}$, $x \in [1, L-\Delta]$. The above expression looks complicated, but all it does is to minimize the variance of each diagonals in distance matrix $\mathbf{D} \equiv \{D(x, x')\}$ for fulfilling the TI requirement. The third loss is interpreted as a biological constraint that puts a large penalty on high firing rate. It is therefore favor solutions of dense code.

$$L3 = \frac{1}{L^2} \sum_{i,x} z_i^4(x) \tag{2.57}$$

V. Training objective. To simultaneously optimize three objectives, I used the most simple strategy, a weighted sum, to reduce the problem to single train objective.

$$\min_{P,\mathbf{a}} \mathrm{Loss}(\beta) \equiv lr1 \cdot L1(\beta) + lr2 \cdot L2 + lr3 \cdot L3, \tag{2.58}$$

where $lr1$, $lr2$, and $lr3$ are the relative learning rates. *VI. Constraint on population firing rate.* At last, there is a single constraint to keep the code bounded: the constraint on total firing rate:

$$\sum_{i,x} z_i^2(x) = L \tag{2.59}$$

Why using a "correlated noise" term to control denoising capability?

Before going into the results from the above optimization scheme, we should first address the question of having a correlated noise term. From the earlier simple two random numbers example, we know that SNR for N identical tuning curves can be boosted up by a factor of $\sqrt{N}$. For a code with large N, SNR scales virtually without bound. This shows just how effective the linear denoising model we adopt in the scheme can be. Unfortunately another force—that increases capacity by diversifying tuning frequencies—is now totally overwhelmed given that the largest Euclidean distance for a pair of codewords is at most 2 (when they are on the opposite side of the unisphere). In other words, the average Euclidean distance is bounded regardless how judicious a combination of frequency is chosen.

The role of the correlated noise term thus becomes clear. It forces a lower bound on total noise term: $\sigma_i^2 \sim \beta/\sqrt{N}$ such that SNR saturates at $\sqrt{N}/\beta/2M$. That is, SNR can only go so far and is bounded by β[22]. After SNR saturates, further increase of redundancy wouldn't help, so it is better to diversify tuning curve frequency for the unused cells—to increase the number of modules.

Convergence to a random set of frequencies with a conserved number of modules

Figure 2.13 shows an example of optimized solutions. The final tuning curves are clearly modular periodic (6 modules from the spectrum). The simulation hyperparameters are listed in Appendix A.10. Figure 2.14 are results from the same hyperparameters but with different random initial coefficient matrix P. Figure 2.14(a) shows that 1) for a fixed denoising capability, $1/\beta$, different random initial conditions converge to the same number of modules with minor deviation, and 2) the final frequencies of a solution are largely at random, i.e. there is no special frequency for a better grid cell code.

Tendency towards higher frequencies

Figure 2.14(b) shows 1) broadly covered cumulative spectra over 100 different random initial P and 2) a uniform phase coverage[23]. The resultant cumulative spectra

[22]Large β: no denoising capability. Small β: strong denoising capability.

[23]The reason for uniform phase coverage in Scheme A-1 is not due to the need for uniform spatial encoding. Since a single pair of sine and cosine tuning curves already cover uniformly entire spatial

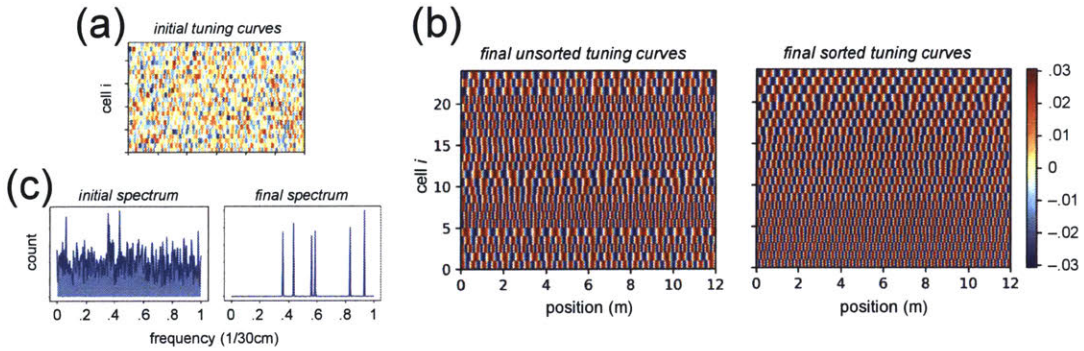

Figure 2.13: An optimized solution converges to a set of sinusoidal tuning curves with few modules. **(a)** Random initial tuning curves. **(b)** The unsorted, and sorted final set of tuning curves revealing its modular periodic nature with uniform phase coverage. **(c)** Left: random initial spectrum; right: modular final spectrum.

reaffirms the statement of no special frequency that echoes the Loss 1 map from 4- or 6-cell TI codes in Figure 2.12. It also shows a tendency toward using higher frequencies as a result of combining the following two effects:

1. Avoiding high Loss 1 in regions with very low frequencies due to the boundary effect of a finite environment.

2. A tendency of collective movement in spectrum between neighboring frequencies—caused by higher overlaps of their tuning curves—towards higher frequency region during the early stage of gradient descent, and before a final winner-take-all (WTA) type of process dominates to select only a few frequencies[24].

Optimal number of modules as a function of denoising capability

To demonstrate how an optimal solution ending up to a certain number of modules for balancing the tradeoff, I ran algorithm with 10 random initial P for each β, and swept β covering the regime from very high-denoising capability to no denoising capability at all. As shown in Figure 2.15(a,b), the larger the denoising capability, $1/\beta$, the smaller the optimal number of modules M—or a more redundant code for fighting

range, the real reason for uniform phase coverage here is because of the phase freedom discussed in Appendix A.1

[24]The second effect largely depends on the variance of the initial amplitudes of P—also in spectrum. The larger the variance, the quicker the WTA-type process resolves the final few frequencies, and the more spread out the final frequencies can be.

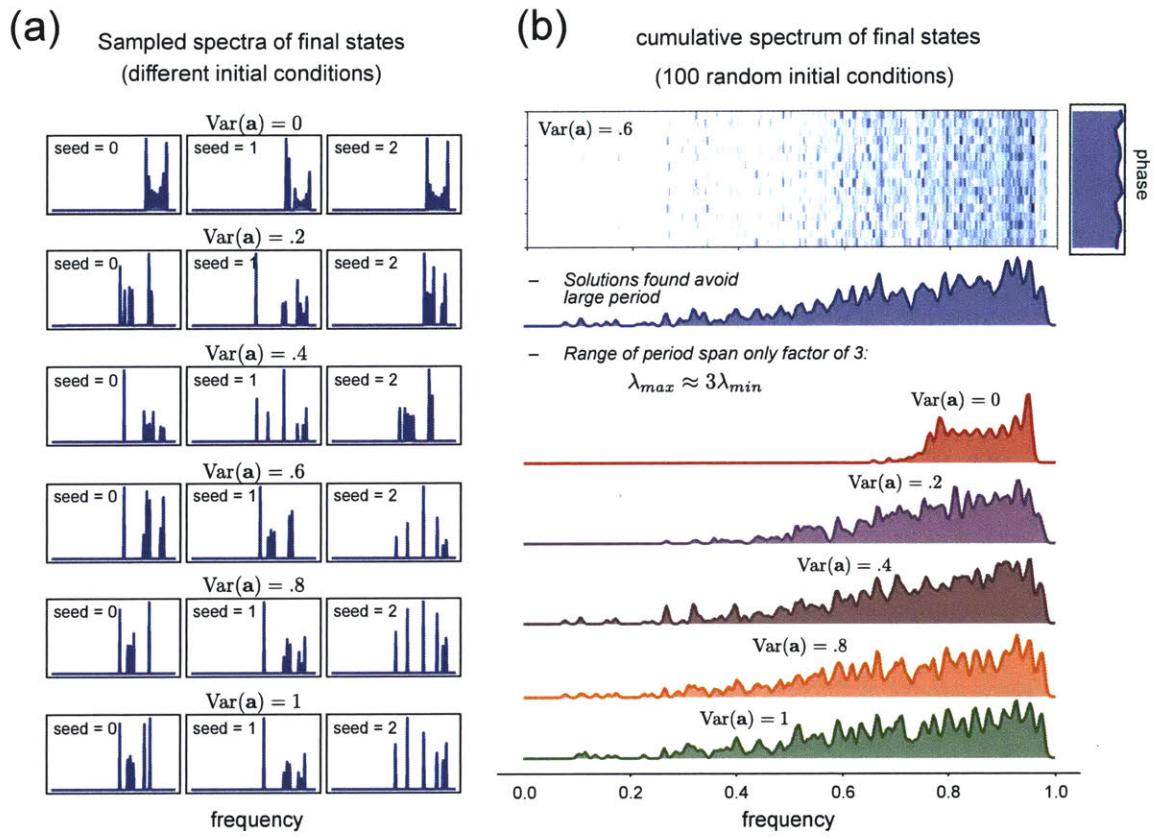

Figure 2.14: Statistics of final spectra indicates that a good solution has no special frequency combination. **(a)** Examples of final spectra from different initial conditions. A final frequency combination is largely at random scattered across higher frequency region. **(b)** Cumulative spectra provide a clear demonstration of 1) no special frequency combination, 2) tendency towards higher frequencies, and 3) uniform phase coverage.

noise. If there is no limit on the denoising capability, i.e. $\beta = 0$, it is always more advantageous to use a single module for boosting SNR.

Next, we can derive an analytics for this tradeoff for further understanding the precise scaling of SNR and average Euclidean distance with respect to N, M, and β. Because in this scheme, SNR does not depend on animal's position (see Equation (2.52)), the distance between normalized codewords can be, thereby, decomposed into a product:

$$D\left(\tilde{\mathbf{z}}(x), \tilde{\mathbf{z}}(x')\right) = D(\mathbf{z}(x), \mathbf{z}(x')) \cdot \frac{1}{\sigma^2}, \tag{2.60}$$

where σ is the average noise over cells. The above assumes that all cells approximately share the same noise level. Like mentioned the Euclidean distance between two codewords is bounded, i.e. $D(\mathbf{z}(x), \mathbf{z}(x')) < 2$, and monotonically increases with

M that can be well fit using a tanh function as shown in Figure 2.15(c).

$$D_{optm}(M) \approx a \ \tanh(bM), \tag{2.61}$$

And SNR scales as

$$SNR \propto \frac{1}{\sigma^2} \sim \frac{2M}{N} + \frac{\beta}{\sqrt{N}} \tag{2.62}$$

The optimal number of module is then to find the maximum of the following equation

$$D\left(\tilde{\mathbf{z}}(x), \tilde{\mathbf{z}}(x')\right) \approx D(M \mid N, \beta) \propto \tanh(bM) \cdot \left(\frac{2M}{N} + \frac{\beta}{\sqrt{N}}\right) \tag{2.63}$$

In Figure 2.15(d), I numerically found optimal M for various β with a fixed N. One can see a qualitatively correct tradeoff following the same trend as that of the full optimization problem—Figure 2.15(a).

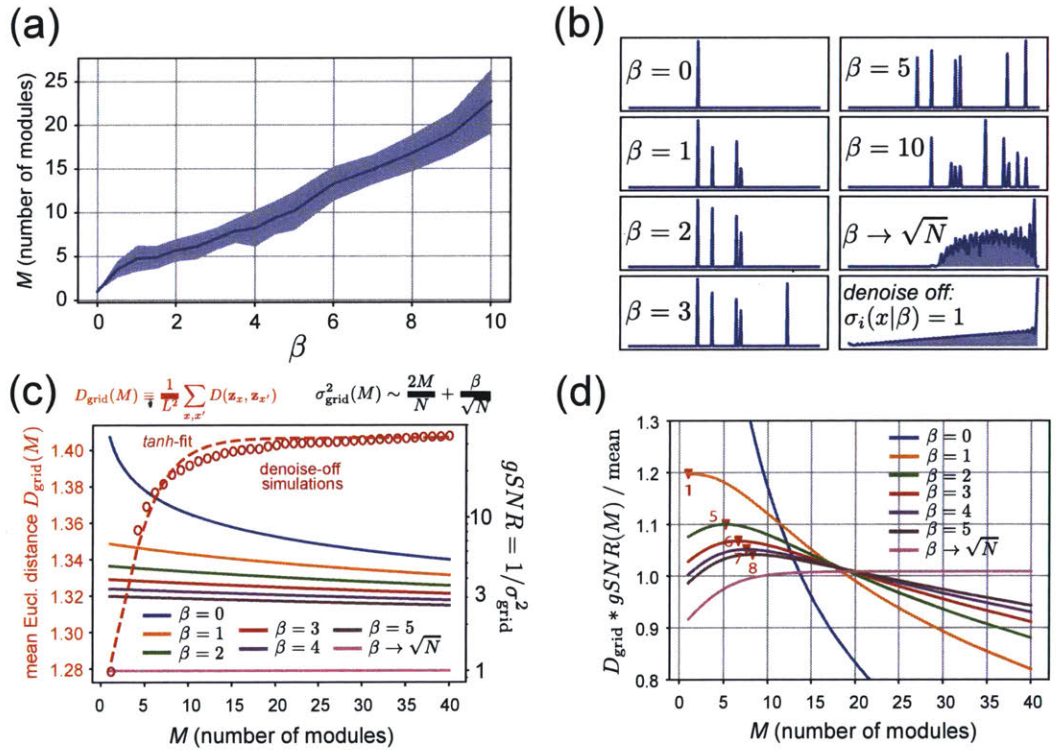

Figure 2.15: Optimal number of modules decreases with increasing denoising capability. **(a)** Optimal number of modules M increases roughly linearly with inverse denoising capability β. **(b)** Example spectra of (a). **(c)** The capacity as average Euclidean distance can be analytically decoupled into a product of a monotonically increasing and decreasing functions of M, i.e. 1) a signal-only distance: D_{grid} and 2) a gain in SNR: g_{SNR}. **(d)** Maxima of the analytical capacity qualitatively predict the numerical optimal M for various β.

Test necessity of each loss function

So far we have seen that Scheme A-1 gave rise to modular periodic tuning curves as optimal solutions. However this demonstration is only halfway through in the optimization principle approach introduced in Chapter 1—it shows that all three loss functions together is sufficient for the emergence. The other half is to demonstrate the necessity of the individual loss functions. To do this, I selectively turned off the loss functions, and observed deviation from the target tuning curves. Figure 2.16(a-c) sum up the necessity for all three loss functions. That is, the target tuning curves only emerge as an optimal solution when all three loss functions are present.

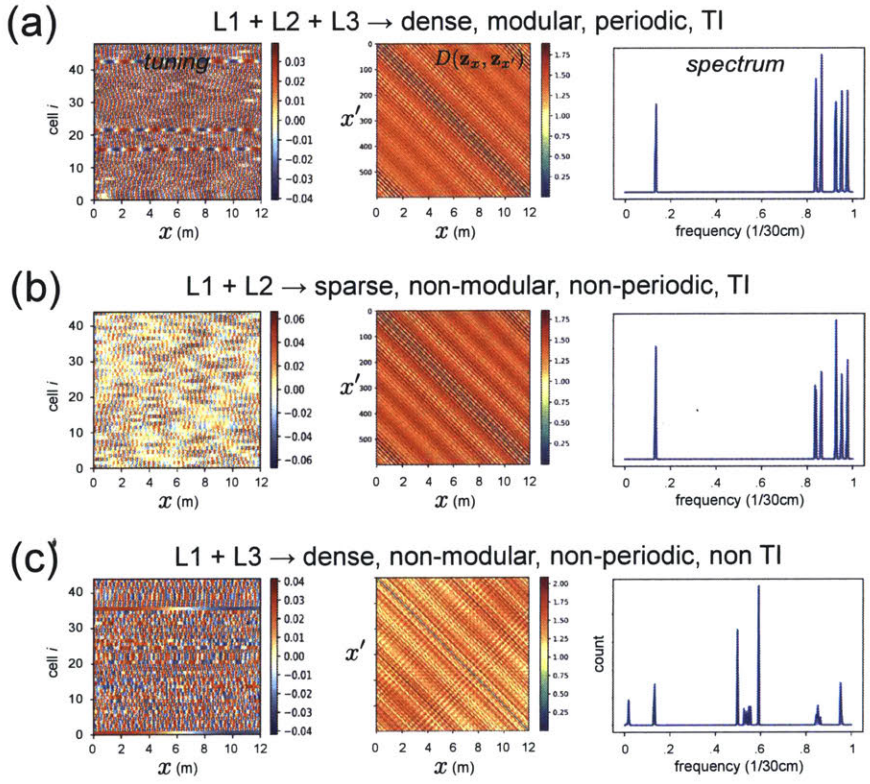

Figure 2.16: All three loss functions are necessary for the emergence. **(a)** Optimal tuning curves possess all features of the target tuning curves when all three loss functions are present. **(b,c)** Optimal tuning curves are not the target tuning curves if one of the three loss functions is turned off.

Advantage of using special frequency ratio between adjacent modules?

There is one seemingly important feature of grid cells observed in experiments—wasn't considered in the target tuning curves—that is, the frequency ratios of the adjacent

modules are roughly a constant (Stensola et al., 2012):

$$\frac{f_i}{f_{i+1}} \approx 1.4 \tag{2.64}$$

From the discussion above, we know already that Scheme A-1 does not lead to an optimal solution with a set of special frequencies. But just for a sanity check, I applied a frequency mask—for selecting a set of frequencies with constant ratio in basis functions—in the original optimization problem. The results are shown in Figureconst freq ratio(a). It is clear that those solutions with constant frequency ratio is not optimal in our scheme, and the optimal solutions do not cluster into certain frequency ratios—Figureconst freq ratio(b). This demonstration implies that whatever function a grid cell code may implement by having a constant frequency ratio is not within our functional hypothesis—which states that a grid cell code has only two main functions: 1) a high-capacity encoding and 2) a robust path-integration.

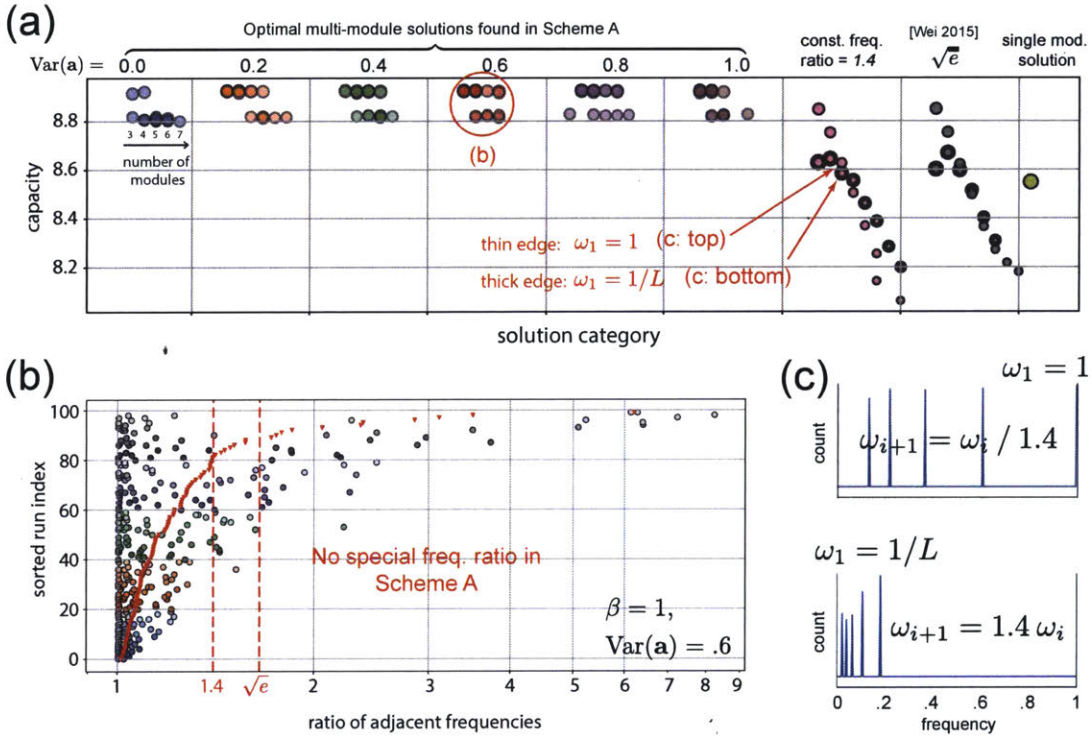

Figure 2.17: A solution with a constant frequency ratio of adjacent modules is not optimal. **(a)** A comparison in capacity across different solution types. Overall, a solution with a fixed frequency ratio has lower capacity. **(b)** Examples of frequency ratios for a set of optimal solutions. **(c)** Two examples of non-optimal solution with a constant frequency ratio.

Summary on advantages of a coding theoretic approach

Below is a list to sum up the finding from Scheme A-1, it emphasizes what a coding theoretic approach offers whereas a direct RNN training approach fails to.

- *A denoising model can be directly implemented and analytically understood in a coding theoretic approach.* The denoising model qualitatively explored the tradeoff between either diversifying tuning curve frequency or grouping them to boost SNR.

- *Coding theoretic approach gives us a bird's-eye view of a much broader energy landscape of an optimization problem.* Since the optimization progressed without involving RNN dynamics, a single run took much less computation cost. Acquiring statistics from a large number of runs became possible.

- *Necessity of either a functional or a biological constraint can be directly tested simply by turning off their corresponding loss functions.* In the earlier demonstration, we saw that all three loss functions are necessary for the emergence of grid cell code.

- *Coding theoretic approach provides a flexible platform to test an alternative hypothesis.* If one wants to explore the effect of an extra condition or constraint on loss function, one can simply apply those on a code. For example, I added a frequency mask to restrict the search region of solutions with a constant frequency ratio. As a result, I found that the restricted solutions were significantly less in capacity. This demonstration showed how a coding theoretic approach can achieve such a level of clarity.

Remaining issues of the ad hoc denoising model

The correlated noise term in the current denoising model is unsatisfactory for two reasons:

1. The linear sum of correlated noise should also depend on the Fourier coefficient matrix just like the sum of uncorrelated noise. E.g. a poisson correlated noise model should sum as $\sum_j P_{ij}^2 \, b_j^2(x)$. Scheme A-1, however, used a constant term merely to lower-bound the noise.

2. A proper denoising model—even with only uncorrelated noise—should already display a correct tradeoff in the benefit of using redundant code.

For these reasons, we will next set up an improvement, Scheme A-2, and discuss in the end the overall plausibility of Scheme A and the denoising model.

2.4.4 A denoising scheme that implements tradeoff correctly

In this section, I will address the issues raised by Scheme A-1 regarding the use of ad hoc correlated noise term. In this improved scheme, I used a more straightforward denoiser with an additional layer of cells. We will see how a complex capacity measure is necessary in order to circumvent the issues raised and, in general, implement a correct tradeoff in such a denoising model.

Setup of Scheme A-2

I. Building tuning curves using Fourier basis functions. The construction of tuning curve is now simpler and without a diagonal matrix **a** as a frequencies picker.

$$z_i(x) = \sum_{j=1}^{N_b} P_{ij}\ b_j(x), \tag{2.65}$$

where $i = 1, ..., N$, $x \in [0, L]$, and $b_j(x) \equiv \cos(2\pi f_{u_j} x + \phi_{v_j})$

with $\begin{cases} u_j & = 1, ..., U \\ v_j & = 1, ..., V \\ j & = 1, ..., N_b \equiv UV \end{cases}$. The frequencies and phases are equally divided: $f_{u_j} \in [0, 1/w]$ and $\phi_{v_j} \in [-\pi, \pi]$.

II. Denoising model. The noise in this denoising model is added to the level of tuning curves.

$$z_i'(x) = z_i(x) \cdot (1 + \xi_i(x)) \tag{2.66}$$

The model is conceptually simpler, however, consequently, we need to introduce a second layer of cells as a denoiser:

$$y_i(x) = \sum_j Q_{ij}\ z_j(x), \tag{2.67}$$

where $Q_{ij} \equiv \frac{\sum_x z_i(x)\ z_j(x)}{|z_i(x)|\cdot|z_j(x)|}$ measures the similarity between tuning curves $z_i(x)$ and $z_j(x)$. As a result, this denoiser sums up similar tuning curves from the layer below and output denoised version of the original tuning curves. The noise σ is summed up differently:

$$\sigma_i^2(x|a_\xi, a_\zeta) = a_\xi^2 \sum_j Q_{ij}^2\ 1_j(x)/N + a_\zeta^2 \sum_j Q_{ij}^2\ z_j^2(x), \tag{2.68}$$

where a_ξ and a_ζ specify the strength of uncorrelated and correlated noises respectively. In the following simulation, I set $a_\xi = 1$ and $a_\zeta = 0$ for simplicity and clarity. We will see that with uncorrelated noise term alone, the denoising model implements correct tradeoff.

III. A capacity measure involving separability and coding dimensionality. Since the denoiser is now at the level of $\mathbf{y}$, the following normalized codewords is used to computed a new capacity measure.

$$\tilde{y}_i(x) = \frac{y_i(x)}{\sigma_i(x)} \cdot \frac{1}{\sqrt{N}} \tag{2.69}$$

The capacity is defined as

$$C(\delta) \equiv S(\delta)\ d^D, \tag{2.70}$$

where

$$S(\delta) \equiv \frac{1}{L^2} \sum_{x,x'} 1 - exp\left(-d_{xx'}^2/\delta^2\right) \tag{2.71}$$

$$d \equiv \left(\prod_{x,x'} d_{xx'}\right)^{1/L^2} \tag{2.72}$$

$$D \equiv \left(\frac{1}{NL} \sum_{i,x} \frac{y_i^2(x)}{\sigma_i^2(x)}\right)^{-1} \tag{2.73}$$

with $d_{xx'} \equiv \sqrt{\sum_i \left(\tilde{y}_i(x) - \tilde{y}_i(x')\right)^2}$ as Euclidean distance between codeword $\tilde{\mathbf{y}}(x)$ and $\tilde{\mathbf{y}}(x')$.

IV. Three loss functions. The geometric mean definition of the separability in Equa-

tion (2.72) allows a simpler form of Loss 1:

$$\begin{aligned} L1(\delta) &\equiv -\log\, C(\delta) \\ &= -\log\, S(\delta) - D\frac{1}{L^2}\sum_{x,x'} \log\, d_{xx'} \end{aligned} \tag{2.74}$$

The second loss function for acquiring a TI code is not changed:

$$L2 = \frac{1}{L}\sum_{\Delta=0}^{N-1} \mathrm{Var}(\mathbf{D}(\Delta)), \tag{2.75}$$

where $\mathbf{D}(\Delta) \equiv \{D(\mathbf{z}(x), \mathbf{z}(x+\Delta))\}$, $x \in [1, L-\Delta]$. Neither is the third loss function:

$$L3 = \frac{1}{L^2}\sum_{i,x} z_i^4(x) \tag{2.76}$$

V. Training objective. follow the earlier approach, I used a weighted sum to reduce the problem to single training objective. The only difference is that there is only P matrix to be optimized.

$$\min_P \mathrm{Loss}(\delta) \equiv lr1 \cdot L1(\delta) + lr2 \cdot L2 + lr3 \cdot L3, \tag{2.77}$$

where $lr1$, $lr2$, and $lr3$ are the relative learning rates.

VI. Constraint on population firing rate. The constraint on total firing rate remain the same:

$$\sum_{i,x} z_i^2(x) = L \tag{2.78}$$

2.4.5 A capacity involving coding dimensionality is needed for leveraging tuning diversity

From the previous denoising model, we know that without an ad hoc correlated noise term for controlling denoising capability, an optimal solution always converged to the ones of single module. This is due to an imbalanced scaling of two counter forces trying to improve overall capacity. As a result, the force from the denoiser—utilizing redundancy to improve SNR—always won. Thus single-module solutions were optimal.

To fix this imbalanced scaling from the denoiser, mathematically, one may consider

a capacity measure that also takes coding dimensionality into account—as introduced:

$$C(\delta) \equiv S(\delta)\ d^D$$

$D_{max} = 1$ specifies full dimensionality when all tuning curves are maximally dissimilar, and $D_{min} = 1/N$ specifies minimal dimensionality when all tuning curves are identical. d^D explicitly implements an exponentially large coding volume among codewords as a function of coding dimensionality. $S(\delta)$ discounts those volume between codewords that cannot be distinguished under a certain noise level δ. Below I show how this new capacity measure implements a correct tradeoff such that a solution with few modules is optimal for a moderate noise level—parameterized as δ.

Convergence to few-module solutions with just uncorrelated noise

From the numerical results with only uncorrelated noise, we can see that the optimal solution reliably converge to that of few modules in Figure 2.18(a). I proceeded to run the algorithm for 100 different random initial conditions, the cumulative spectrum is plotted in Figure 2.18(b-d). The optimal solutions again cover all frequency range relatively uniformly without a special frequency. Curiously, this time there is a more pronounced boundary effect that causes a frequency-phase lock manifesting four tilted stripes in frequency-phase histogram. Also this time there was no obvious tendency toward higher frequency solutions.

Scheme A-2 implements tradeoff correctly

Next, By sweeping the noise level we can see manifestations of a correct tradeoff in Figure 2.19:

1. Number of modules M decreases with increasing noise level δ.
2. Capacity C decreases with increasing noise level δ.

Figure 2.19(b-d) shows three examples of different noise level representing qualitatively the same features of overall 100 solutions in Figure 2.19(a).

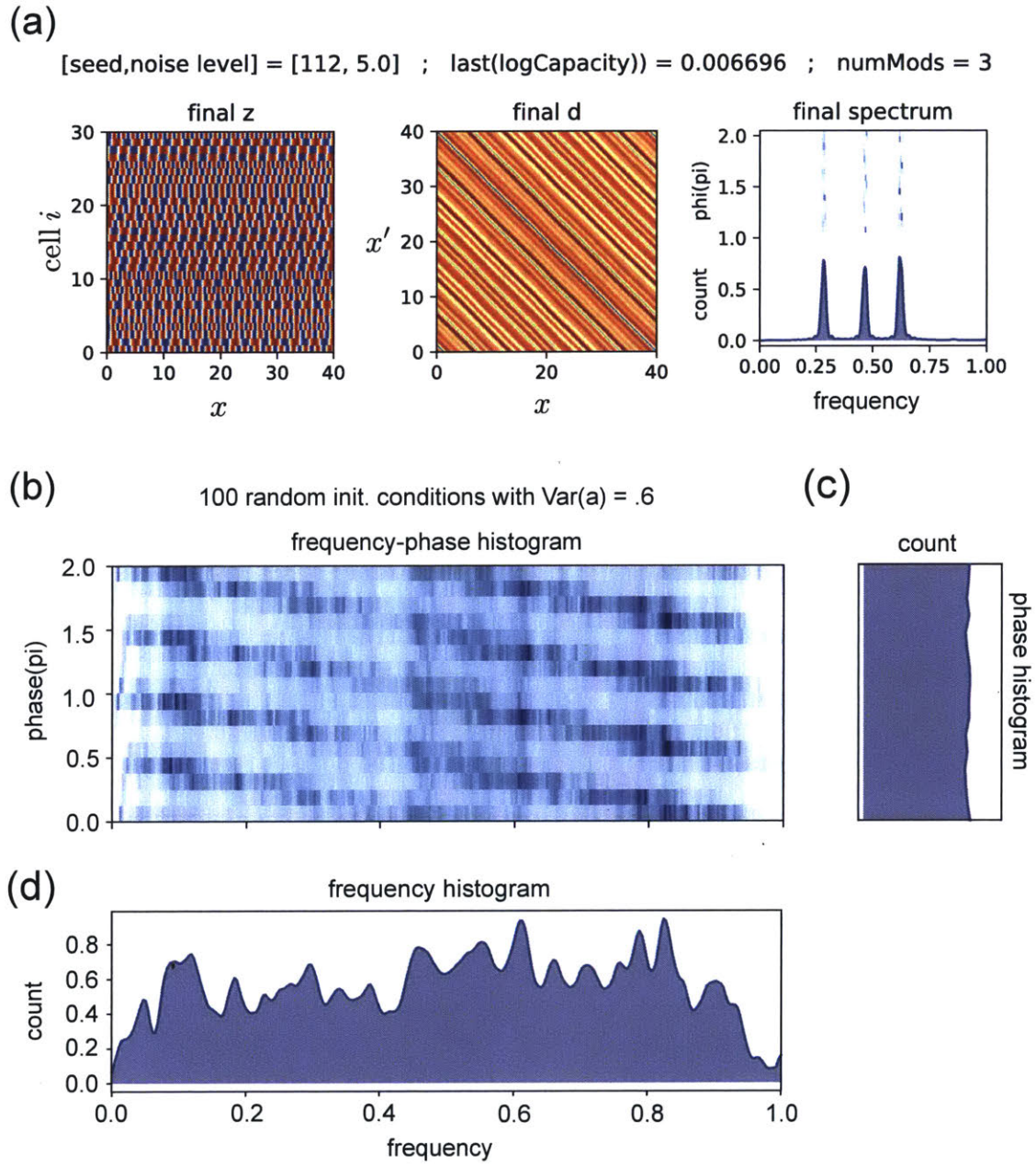

Figure 2.18: Optimal solutions in Scheme A-2 shares the same tuning properties with Scheme A-1. **(a)** An optimized solution with moderate noise level coverage to a 3-module sinusoidal tuning curves. **(b)** Cumulative frequency-phase histogram of 100 random initial conditions. **(c)** Uniform phase coverage. **(d)** Uniform coverage in frequency reaffirms no special frequency combination for an optimal solution.

Remaining issues in general

The attempt of Scheme A-2 to address the issues—regarding the ad hoc denoising model from Scheme A-1—revealed some fundamental problems of the need for a

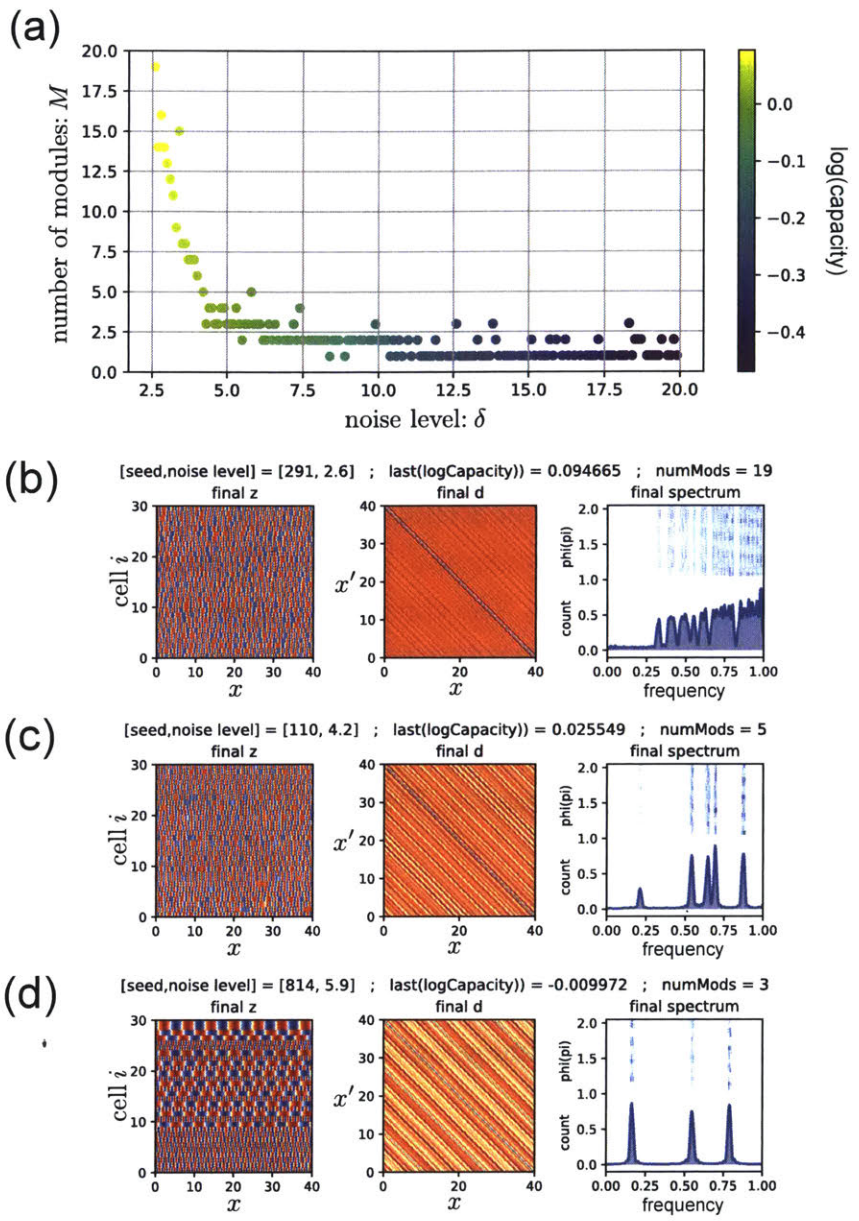

Figure 2.19: Scheme A-2 implements correct tradeoff so that optimal number of modules decreases with increasing noise level. **(a)** Both optimal number of modules and capacity decreases with increasing noise level. **(b-d)** Three examples with different noise levels.

denoiser.

I. Three noise-related parameters Ideally, there should be only one noise in a system for demonstration purposes. This scheme, however, required three: a) an intrinsic noise set by maximum frequency $1/w$ in basis functions, b) an uncorrelated noise a_ζ in the denoiser, and c) a threshold noise δ that discounts ambiguous codewords. In principle, these three noises should not be, independent, free parameters. They

should, instead, be connected via a unified underlying noise assumption. In that attempt, one might introduce and fine-tune few more parameters for such a potentially elaborate noise model. And it could obscure the purpose of demonstration. To avoid this, I fixed the first two parameters: w and a_ζ, and only vary δ for clarity. Nonetheless, the issue of three-noise assumption needs to be addressed properly.

II. Unlikely complex capacity measure Besides the implausible involvement of three noises, our only working capacity measure involves computing an unlikely global quantity: coding dimensionality. Computing coding dimensionality requires tracking activities of N cells over a long firing history L. It's difficult to imagine the biology will try to optimize a design based on computing such elaborate statistical quantity using yet another circuitry. Most importantly, while it is the fact that the average coding volume between codewords scales as d^D, this exponential scaling only provides more space to fit more codewords in. The ability to distinguish all existing codewords should still scale with Euclidean distance alone. The bottom line is that this capacity measure is, again, an ad hoc mathematical fix—which is not backed by a sound theoretical motive. And for this reason, we should be critical about the plausibility of the very first assumption of Scheme A: The Use of sinusoidal functions as target tuning curves.

2.4.6 Remarks of Scheme A

- Scheme A targets sinusoidal tuning curves, and a denoiser is thereby needed for module formation.
- Scheme A demonstrates how noise plays an important role in balancing degree of redundancy and thus shaping the optimal solution. Such fine balance of optimal solution leads to a noise-dependent optimal number of modules.
- Scheme A shows how coding theoretic approach is advantageous in various aspects compared to RNN training approach—especially when it comes to affirming the necessity of hypothesized functions of a system.
- Scheme A uses a necessary—but unlikely—complex capacity measure for implementing a correct tradeoff that leads to few-module solutions. The unlikeliness of the capacity measure forced one to reconsider the plausibility of the first assumption—*sinusoidal* target tuning curves.

2.5 Scheme B: Sparse code with optimized separability

The main takeaway from Scheme A regarding using *sinusoidal* target tuning curves is: a mechanism for module formation—that is a key feature of grid cell codes—is absent. Consequently, an implausible extra denoiser as well as a complex capacity measure needed to be implemented. To resolve this issue, we need a much simpler scheme. In Scheme B, I will remove the need for a denoiser by targeting a different, more sparse tuning curve. We will see how this scheme can potentially give rise to a modular tuning curves regardless of having no external denoising model and, towards the end, why a new capacity measure instead of average Euclidean distance—that measures how separable the codewords are—is needed. Lastly, for clarity in comparison with Scheme C in the next section, I will call our capacity measure—*separability.*

2.5.1 Simple noise assumption and an implicit sparse constraint

The target tuning curves in Scheme B resemble more closely—than that of Scheme A—to the experimentally observed grid cell tuning curves, which are well described by von Mises functions:

$$z_i(x) \propto \exp\left(\frac{\cos(2\pi f_i x + \varphi_i) - 1}{\sigma_i^2}\right). \tag{2.79}$$

A von Mises tuning curves has three important features: 1) it is locally a narrow gaussian function with standard deviation σ_i, 2) it is globally periodic, and 3) it is positive. Features 1) and 3) contrast the earlier target tuning curve. Because of these new features, it is now possible to have a module formation mechanism without external denoising model—as will be discussed later. I will, first, demonstrate how the new tuning features 1) and 3) are in fact related under capacity maximization.

This connection can be derived using a connection—in a population code—between Euclidean distance and autocorrelation. Note that, the average circular autocorrelation over all cells is

$$\langle c \rangle \equiv \frac{1}{L}\sum_i \sum_x c_i(x) = \frac{1}{L}\sum_i \sum_x \left(\frac{1}{L}\sum_{x'=0}^{L} z_i(x)\, z_i(x')\right), \tag{2.80}$$

which essentially measures long-range tuning similarity of each cells. On the other hand, the average square Euclidean distance is given as

$$\begin{aligned}\langle d^2 \rangle &\equiv \frac{1}{L^2} \sum_{x,x'} \left(\sum_i (z_i(x) - z_i(x'))^2 \right) \\ &= \frac{1}{L^2} \sum_{x,x'} \left(\sum_i z_i^2(x) + z_i^2(x') - 2 z_i(x) z_i(x') \right) \\ &= 2 - 2 \frac{1}{L^2} \sum_{x,x'} \sum_i z_i(x) z_i(x') \\ &= 2\left(1 - \langle c \rangle\right)\end{aligned} \tag{2.81}$$

The above simplification uses an assumption of unity codeword length : $\sum_i z_i^2(x) = 1$. It shows that maximizing $\langle d^2 \rangle$ is equivalent to minimize $\langle c \rangle$. It also tells us that maximizing a population quantity $\langle d^2 \rangle$ can be understood by just minimizing a single cell quantity, $\frac{1}{L} \sum_x c_i(x)$, the average autocorrelation of cell i.

Meanwhile, we note that the minimal single cell autocorrelation can be achieved via using a white noise tuning curve. The autocorrelation scales as $c_i(x) \propto e^{-x/\sigma}$, where σ—sets the maximal spatial resolution—is small and hence a fast decaying $c_i(x)$. The reason that the autocorrelation quickly decays to zero for large offset is due to the cancellation between random positive and negative signals of white noise. Now if we constraint tuning curves to be strictly positive, the only way to achieve the same minimal autocorrelation is to use a sparse tuning curve: $z_i(x) \propto e^{-x^2/2\sigma^2}$.

The resultant sparse codes echoes the sparse representation in V1 in primate visual system. The sparse representation in the brain was originally hypothesized as a strategy to maximize information about stimuli (Bell and Sejnowski, 1995). The equivalence between sparseness and maximal information was demonstrated by maximizing output information about the stimulus for nonnegative neurons (van Vreeswijk, 2001). The resultant representation was sparse even though sparseness was not imposed as an additional constraint.

From above we see how a nonnegativity constraint—an innate property of a real cell's tuning curve—naturally leads to sparse tuning curves. This makes this candidate biological constraint for Loss 3 ever more plausible because we don't even have to put in sparseness constraint explicitly. To conclude, a high capacity code with positive tuning curves is naturally sparse.

2.5.2 Robustness-separability tradeoff

Before moving on to the full optimization problem, we first use Scheme B-1 to understand within which parameter regimes a multi-field solution—a key tuning property of grid cells—is optimal. The optimization problem of Scheme B-1 is meant to establish two essential facts:

1. An optimal solution is unimodal given enough cells.
2. An optimal solution displays a robustness-separability tradeoff under various levels of intrinsic noise.

As mentioned, in this optimization scheme I removed the need for a denoiser and introduced a new nonnegative tuning constraint for Loss 3. Other parts of the setup remains the same as Scheme A. The challenge for this simple scheme lies on the issue of approaching global optimum for which I will address in the next section.

Setup of Scheme B-1

I. Simple scheme without denoiser.

$$z_i(x) = \sum_{j=1}^{N_b} P_{ij}\ b_j(x), \tag{2.82}$$

where $i = 1, ..., N$, $x \in [0, L]$, and $b_j(x) \equiv \cos(2\pi f_{u_j} x + \phi_{v_j})$

with $\begin{cases} u_j & = 1, ..., U \\ v_j & = 1, ..., V \\ j & = 1, ..., N_b \equiv UV \end{cases}$. The frequencies and phases are equally divided: $f_{u_j} \in [0, 1/w]$ and $\phi_{v_j} \in [-\pi, \pi]$.

II. Separability as capacity measure. Note that since we don't have a denoiser this time, separability is directly calculated from Euclidean distance between pairs of codewords:

$$D(x, x') \equiv D(\mathbf{z}(x), \mathbf{z}(x')) = \sqrt{\sum_i (z_i(x) - z_i(x'))^2} \tag{2.83}$$

III. First loss functions. The first loss functions is for maximizing capacity as usually

except that it is now averages over D^2 instead of log D as it was in Scheme A.

$$\begin{aligned} L1 &\equiv -\frac{1}{L^2}\sum_{x,x'} D^2(x,x') \\ &= -\frac{1}{L^2}\sum_{i}(z_i(x)-z_i(x'))^2 \end{aligned} \tag{2.84}$$

IV. Second loss functions. We don't use second loss function in this first demonstration.

V. Third loss function. The third loss function is a soft constraint on demanding nonnegative tuning:

$$L3 = \log\ \left(r_z^2 - r_{z,tar}^2\right)^2, \tag{2.85}$$

where the a ratio r_z^2 is defined as negativity of tuning curves $\mathbf{z}(x)$:

$$r_z^2 \equiv \frac{1}{NL}\frac{\sum_{i,x} z_{-,i}^2(x)}{\sum_{i,x} z_{+,i}^2(x)} \tag{2.86}$$

with $\begin{cases} z_{-,i}(x) &= [z_i(x)]_- \\ z_{+,i}(x) &= [z_i(x)]_+ \end{cases}$; $[u]_- = \begin{cases} u & \text{if } u < 0 \\ 0 & \text{if } u \geqslant 0 \end{cases}$ and $[u]_+ = \begin{cases} u & \text{if } u \geqslant 0 \\ 0 & \text{if } u < 0 \end{cases}$. The target $r_{z,tar}^2$ is a chosen and fixed number.

VI. Soft constraint on codeword length. In this scheme I used a soft constraint on demanding unity codeword length. The constraint helps a solution to avoid being trapped in a local minimum that has only few cells firing.

$$L0 = \log\ \frac{1}{L}\sum_{x}\left(1 - \frac{1}{N}\sum_{i} z_i^2(x)\right)^2 \tag{2.87}$$

VII. No single training objective. In this scheme, the energy landscape is rough. Using a single training objective as a weighted sum of all three loss function is no longer effective. As an alternative, we will use a technique I named stochastic orthogonal gradient descent discussed in Appendix A.5.

$$\min_P\ [L0,\ L1,\ L3], \tag{2.88}$$

VIII. Constraint on population firing rate. At last the constraint on total firing rate

remains the same:

$$\sum_{i,x} z_i^2(x) = L \tag{2.89}$$

Using the above setup, I fixed the coding range to be $L = 20$ and swept the number of cells N. The results shown in Figure 2.20. Firstly, we can see that the separability $\langle d \rangle$ saturates after $N \gtrsim 40$. This number depicts a condition of using exactly one cell to encode one location—a place-cell-like code as will be discussed in Chapter 3. The condition of having enough cells can be given as an inequality:

$$N > \frac{L}{\gamma w}, \tag{2.90}$$

where $\gamma = .48$ is a linear-fit parameter from Figure 2.10. If an optimal code has enough cells: $N > 40$, its tuning curves are always unimodal (single field). Secondly, as shown in Figure 2.21, there is a robustness-separability tradeoff for those solutions with enough cells. That is, a more robust code using larger tuning width w leads to a lower separability $\langle d \rangle$:

$$\langle d \rangle_{optm}(w) = \sqrt{2}(1 - \log 4 \cdot \frac{\gamma w}{L}) \tag{2.91}$$

This analytical upper-bound is derived in Appendix A.4.

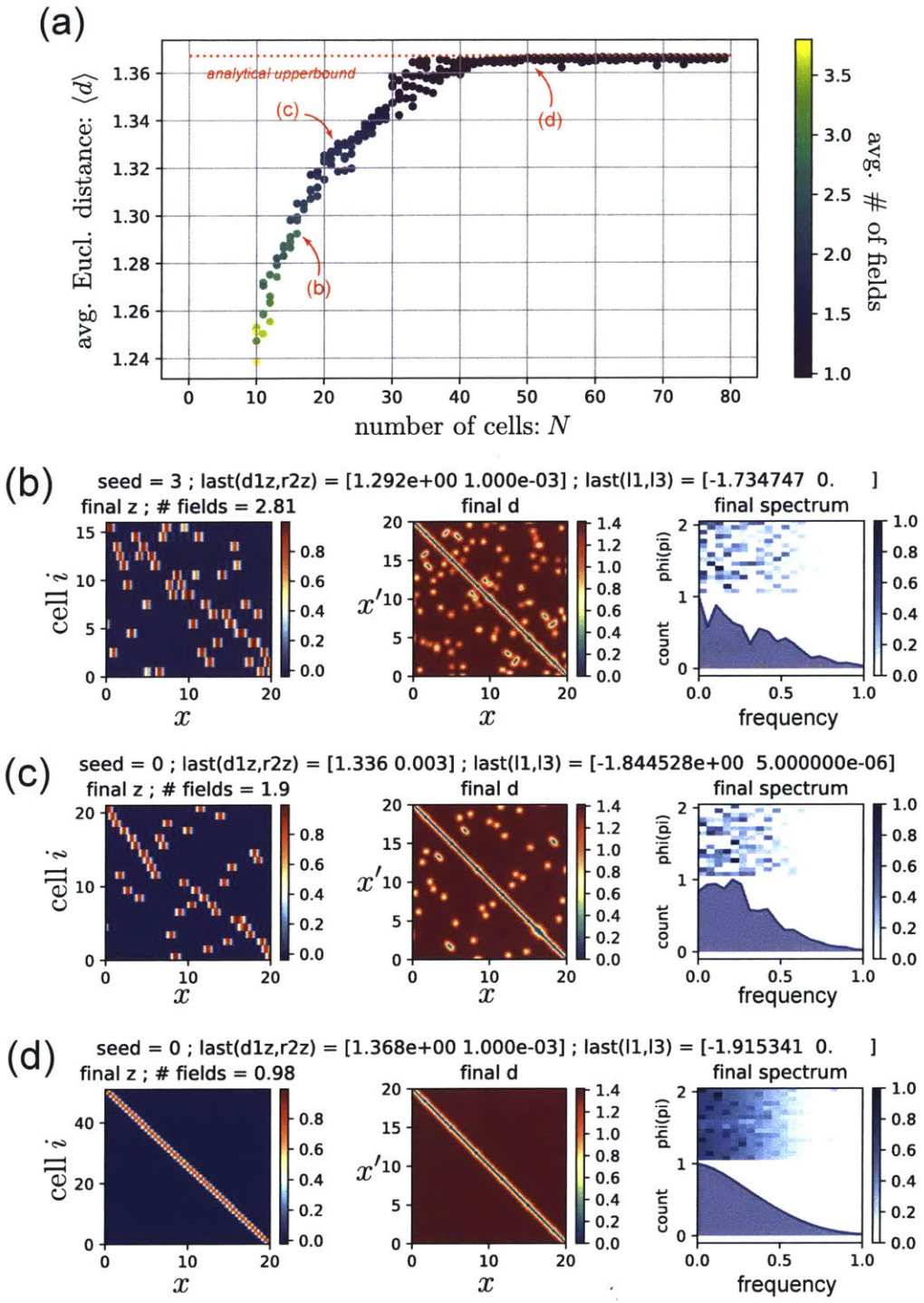

Figure 2.20: An optimal solution is unimodal given enough cells in Scheme B-2. **(a)** With a fixed maximal frequency $1/w$ in Fourier basis functions, an optimal solution has one field per cell if the number of cells, N, satisfies $N > L/\sigma = 20/.48 = 42$. (b-d) Three examples with different N.

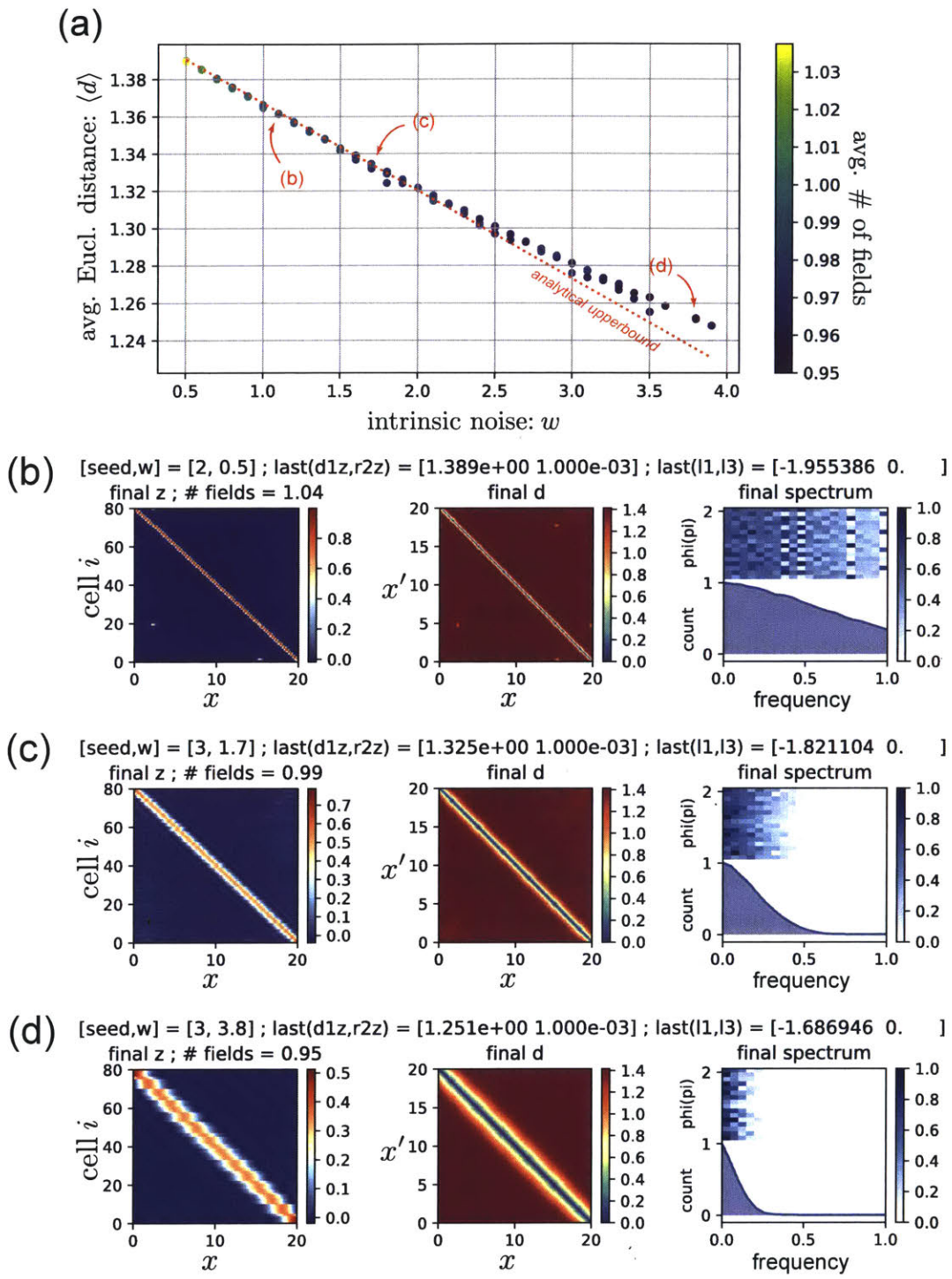

Figure 2.21: Optimal capacity decreases linearly with noise level. **(a)** Given enough cells, an optimal solution has capacity linearly decreases with noise level w. **(b-d)** Three examples with different w.

2.5.3 How modules might emerge without a denoising model?

After introducing the parameter regime for a set of multimodal tuning curves to be the optimal solution, we are in a good place in discussing of module formation mechanism. Specifically it is curious to ask how a scheme lacking of denoising model might still group tuning curves for redundancy. In the current scheme, a tradeoff from the following two opposing forces may lead to module formation:

1. An already existing module wants to recruit more cells for sharpening tuning widths to increase intra-module separability.

2. Population as a whole would like more modules to avoid repeating codewords.

Early on in Scheme A, the above tendency 1 does not exist because the target tuning curves are dense and of sinusoidal functions. But with a new Loss 3 that favor sparse code, the more cells within a module allows for achieving narrower tuning curves (until the width saturates). Module formation therefore may happen when there is not enough cells or when noise level is small: $N < L/\sigma$.

2.5.4 Modular periodic solution is not typical

Scheme B-2 in this section adds back the second loss function that demands a TI code. The challenge in optimization is constant conflicts between three loss functions. Using the method of the stochastic orthogonal gradient descent developed in Appendix A.5, I was able to largely improve the optimization such that both separability S and nonnegativity $1/r_z^2$ increases overall for an optimized solution.

Setup of Scheme B-2

I. Simple scheme without denoiser.

$$z_i(x) = \sum_{j=1}^{N_b} P_{ij}\ b_j(x), \tag{2.92}$$

where $i = 1, ..., N$,$x \in [0, L]$, and $b_j(x) \equiv \cos(2\pi f_{u_j} x + \phi_{v_j})$

with $\begin{cases} u_j & = 1, ..., U \\ v_j & = 1, ..., V \\ j & = 1, ..., N_b \equiv UV \end{cases}$. The frequencies and phases are equally divided: $f_{u_j} \in [0, 1/w]$ and $\phi_{v_j} \in [-\pi, \pi]$.

II. Separability as capacity measure. Separability is calculated from averaging Euclidean distance over pairs of codewords.

$$D(x, x') \equiv D(\mathbf{z}(x), \mathbf{z}(x')) = \sqrt{\sum_i (z_i(x) - z_i(x'))^2} \tag{2.93}$$

III. Loss functions. All loss functions remain the same except I added the second loss function back in this full optimization problem for ensuring a TI code.

$$\begin{aligned} L1 &\equiv -\frac{1}{L^2} \sum_{x,x'} D^2(x, x') \\ &= -\frac{1}{L^2} \sum_i (z_i(x) - z_i(x'))^2 \end{aligned} \tag{2.94}$$

The second loss function now takes a softer form. It first computes a target constant diagonal matrix out of the ongoing distance matrix $D(\mathbf{z}(x), \mathbf{z}(x'))$:

$$D^2_{tar}(x, x') = \frac{1}{L - \Delta} \sum_{x=1}^{L-\Delta} D(x, x + \Delta) \tag{2.95}$$

$D^2_{tar}(x, x')$ replaces the value of each entry in $D^2(x, x')$ to the mean along its diagonal. The second loss function is now simply

$$L2 = \log \frac{1}{L^2} \sum_{x,x'} \left(D^2(x, x') - D^2_{tar}(x, x') \right)^2, \tag{2.96}$$

IV. Third loss function. The third loss function differ from that of Scheme B-1 in a way that it is a soft version of the last. The constraint on demanding nonnegative tuning with a moving target bounded to the final target:

$$L3 = \log \left(r_z^2 - \theta \left(.99\, r_z^2, r^2_{z,tar} \right) \right)^2, \tag{2.97}$$

where $r_z^2 \equiv \frac{1}{NL}\frac{\sum_{i,x} z_{-,i}^2(x)}{\sum_{i,x} z_{+,i}^2(x)}$ with $\begin{cases} z_{-,i}(x) & = [z_i(x)]_- \\ z_{+,i}(x) & = [z_i(x)]_+ \end{cases}$ is the negativity of tuning curves $\mathbf{z}(x)$,

$\theta(u, u_{tar}) = \begin{cases} u & \text{if } u > u_{tar} \\ u_{tar} & \text{if } u \leqslant u_{tar} \end{cases}$ is a clip function, and

the target $r_{z,tar}^2$ is a chosen and fixed number.

V. Soft constraint on codeword length in stochastic orthogonal gradient descent. In this full optimization problem, Loss 0 is not directly optimized but used to compute a corresponding gradient vector used in stochastic orthogonal gradient descent.

$$L0 = \sum_x \left(1 - \frac{1}{N}\sum_i z_i^2(x)\right)^2 \tag{2.98}$$

VI. Multiple objectives. I used stochastic orthogonal gradient descent introduced in last section on three main loss functions.

$$\min_P [L1,\ L2,\ L3], \tag{2.99}$$

VII. Constraint on population firing rate. The constraint on total firing rate remains the same:

$$\sum_{i,x} z_i^2(x) = L \tag{2.100}$$

Modular periodic solutions are not typical

Figure 2.22 shows 100 optimized solutions with vanishing Loss 2, i.e. these solutions are TI codes. There is a clear tradeoff between minimizing Loss 1 and Loss 3, or between maximizing separability and nonnegativity, as shown in Figure 2.22(a). The colored markers indicate the number of modules—computed based on the similarity of a pair of tuning curves (Appendix A.3). There are few remarks to be made from this plot:

1. There is a clustering of optimized solutions towards a potential Pareto front (Shoval et al., 2012).

2. There are only 18 out of 100 optimized solutions that are modular (of more than one module).

3. Modular solutions are less optimal (further away from Pareto front) compared to single module solutions.

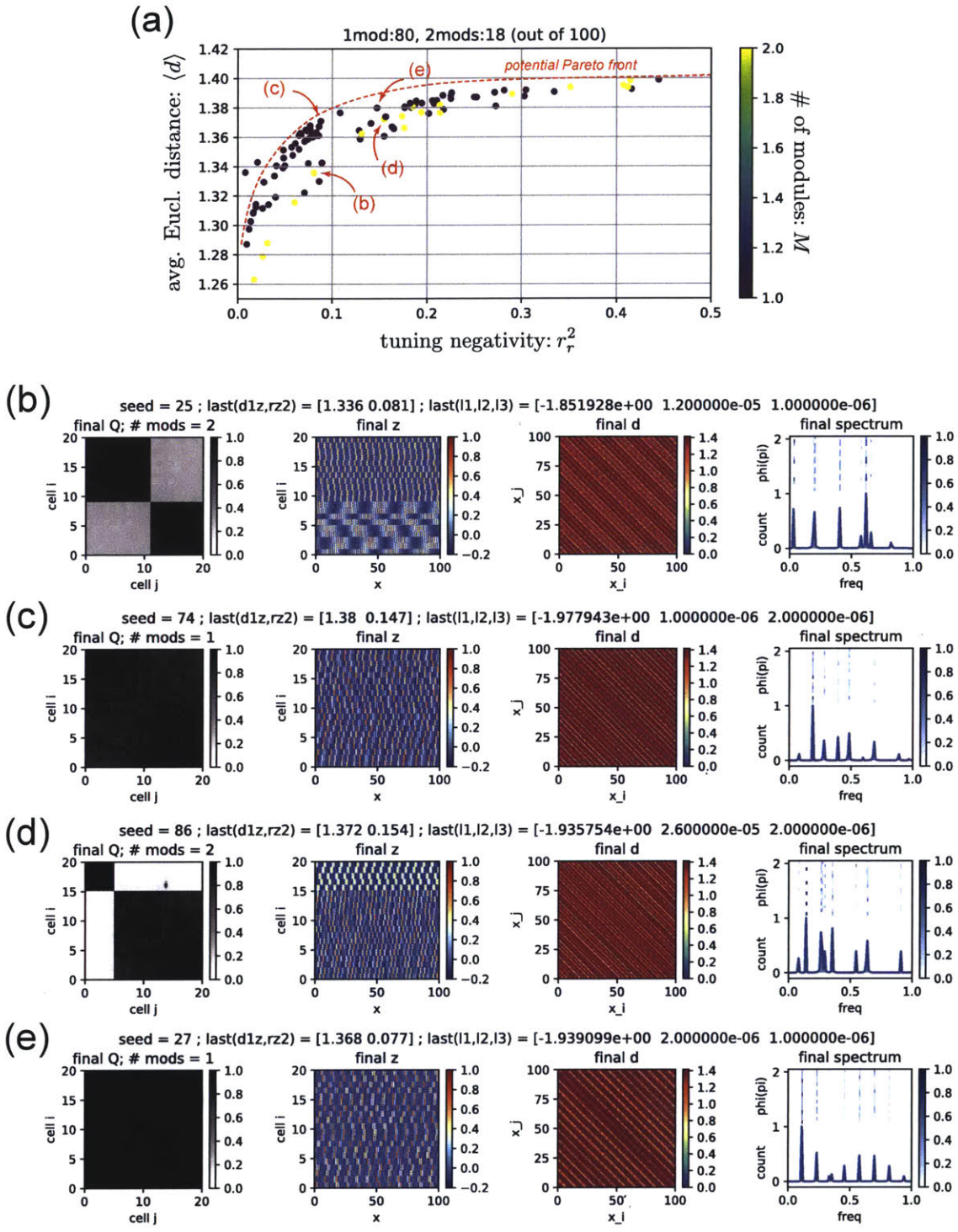

Figure 2.22: Multi-module solutions are not optimal in Scheme B-2. **(a)** A range of optimized solutions cluster near a potential Pareto front with a tradeoff between capacity and tuning nonnegativity. Only a minority of these solutions have more than one modules, which resembles grid cells. **(b-e)** Four example solutions in (a). The leftmost plots are tuning curves similarity matrices used to quantifies number of modules (Appendix A.3).

2.5.5 Remarks of Scheme B

After trying a variety of Loss 3 and attempting to push the solution towards global optimum. Scheme B-2 is by far the closest I can get. However, the optimal solutions are still not grid cell codes. These results forced me to reevaluate the plausibility of the current function hypothesis; more specifically, to reconsider the feasibility of using average Euclidean distance as the capacity measure.

While being aware that a more systematic study needed to be done to make a solid claim, I am convinced from the results of Scheme B that the main objective of a grid cell code is not only to achieve overall high coding separability, but also to have a long unsegmented robust coding line. In the next section, I will introduce a new capacity measure in Scheme C that could be of much more discriminability for narrowing down good solution space and excluding non-grid cell solutions, or more specifically, non-modular solutions as I will explain next.

2.6 Scheme C: A new capacity measure, towards emergence of grid cell code

In this section, I propose another optimization scheme that uses a new capacity measure. Scheme C is mostly the same as Scheme B, yet it no longer uses coding separability for Loss 1. The new capacity measure resembles that we have discussed in the beginning of this chapter: the capacity measure for binary neurons. A notion of threshold noise is thus necessary on top of the intrinsic noise specified by the minimal tuning width (or maximal frequency in basis functions). I will end this chapter by pointing out the remaining issues about reaching the global optimum and suggesting how one may overcome these issues.

2.6.1 New capacity measure demands a long robust coding line

We will start from an example of 4-cell code to motivate the use of new capacity measure with its higher discriminability (higher contrast) on neighboring solutions than that of coding separability defined as average Euclidean distance. In this example shown in Figure 2.23, the frequency of first cell pair is fixed to 1 and the second

frequency is chosen to be either $1/\pi$ for the low-capacity code (top set of 6 plots) or $1/\varphi$ for high-capacity code (bottom set of plots), where φ is the golden ratio. The 4-cell codes are TI by default, an additional random orthogonal matrix was applied to rotate the whole codeword structure to an arbitrary direction in state space. z_1 and z_2 in Figure 2.23(d) thus can be seen as a random projection from a 4D state space onto a 2D plane.

Operational definition of separability. Figure 2.23(b,c) defines separability in the following steps:

1. Plot Euclidean distances of all codewords relative to the first codeword as shown in Figure 2.23(b).
2. Calculate separability by averaging distances over a chosen coding range $x < L$ excluding the neighboring codewords $x < 1/2$ shown as yellow area in Figure 2.23(b).
3. Compute average separability over different integral range shown as blue area in Figure 2.23(c).

Operational definition of new capacity. Figure 2.23(e,f) defines capacity in the following steps:

1. Plot Euclidean distances of all codewords relative to the first codeword as shown in Figure 2.23(e).
2. Calculate capacity by finding the first collision point of a given threshold noise; i.e., $C \equiv \min \arg(d(0,x)) < d_{thres}$, where $x > 1/2$ marked as green star in Figure 2.23(e).
3. Compute average capacity over different threshold noise shown as blue area in Figure 2.23(f).

Discriminability of separability and capacity. Early on in an 4-cell code example in Scheme A (Figure 2.12), we see how a solution is a local maximum in Loss 1 (or local minimum in separability) if a chosen frequency ratio between two modules is a rational number. The example frequencies, $1/pi$ and $1/varphi$, we used here are both irrational numbers. However, the golden ratio *varphi* is a number so called the most irrational number, whereas pi is very close to the neighboring rational number. One

therefore can expect that the case with $f_2/f_1 = 1/\varphi$ should result a larger separability, which is indeed the case: $\langle S_\pi \rangle = 1.349$ vs. $\langle S_\varphi \rangle = 1.351$. However, their separabilities are almost identical, so as all the other frequency choices between them. To better discriminate these two codes. We should observe their overall coding line structures.

In Figure 2.23(1d), we can see that for the case of $f_2/f_1 = 1/\pi$, the coding line is cramped in some areas but loose in others, the unevenness of codeword distribution is eventually compensated and result in a relatively high separability. As for capacity, however, it is very sensitive to the precise relative location, $x - x'$, of those close by codewords. Therefore this uneven coding line has small capacity $\langle C_\pi \rangle = 2.573$ compared to $\langle C_\varphi \rangle = 3.094$ in the case of $f_2/f_1 = 1/\varphi$. The golden frequency ratio has evenly distributed coding line as shown in Figure 2.23(2d) which results in a robust code with its capacity decreases smoothly during a gradual increase of the threshold noise as shown in Figure 2.23(2f).

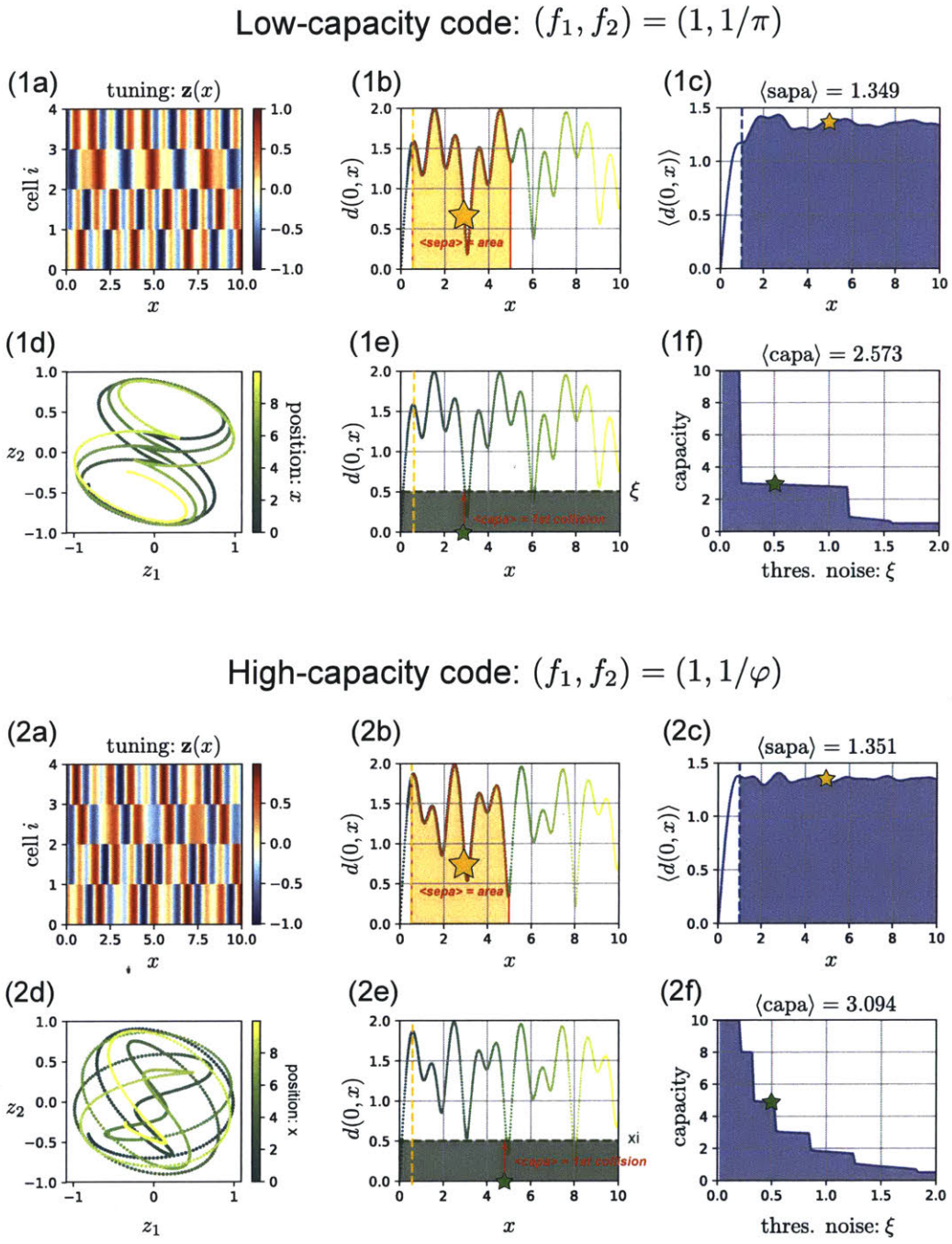

Figure 2.23: A good code based on the new capacity measure has uniformly distributed coding lines. **(1a)** Tuning curves of a low-capacity 4-cell TI code with frequencies: $(f1, f2) = (1, 1/\pi)$. **(1b)** Operational definition of separability as average Euclidean distance over a given coding range. **(1c)** Separability quickly saturates with increasing coding range. **(1d)** Coding line (1D manifold) of tuning curves of (1a). **(1e)** Operational definition of the new capacity as spatial distance of first collision point given a threshold noise ξ. **(1f)** Capacity quickly decreases with increasing ξ (critical failure: a sign of non-robust code). **(2a-2f)** Same but for a high-capacity code with $(f1, f2) = (1, 1/\varphi)$.

Optimal solution regions of separability do not overlap with that of capacity. In

Figure 2.24(a), I plotted both separability and capacity for all frequency choices. The shaded region are relatively bad solutions with strong boundary effect caused by finite coding range. In the middle frequency range: $.6 < f < .9$, I defined the top 30% as good solutions of separability (blue) or capacity (orange) and re-plotted in Figure 2.24(b). First, we can clearly see that capacity has better contrast than separability in discriminating good from bad solutions, and second, their corresponding good solution regions do not overlap much. This result has important implication: the energy landscape and locations of optimal solution from the new capacity measure will be different from that of Scheme B-2 in Figure 2.27 which used separability, and thus has a chance to favor more grid cell like codes.

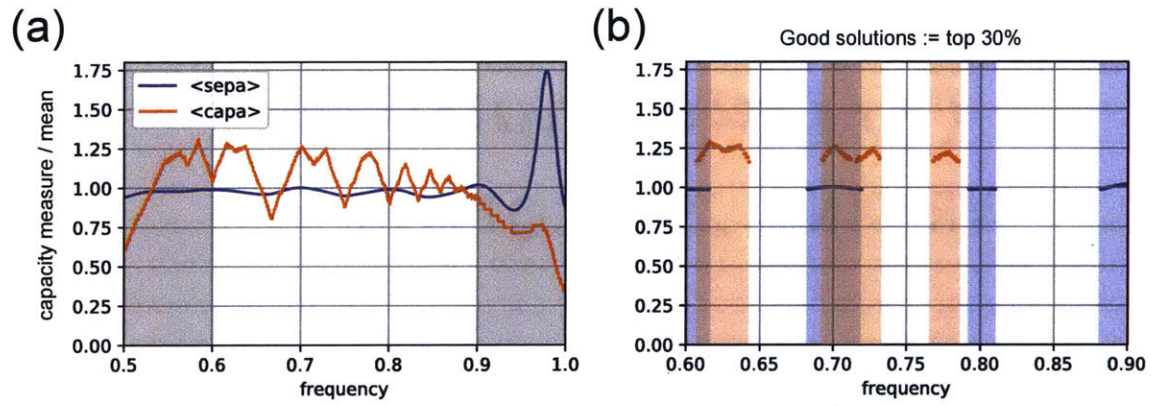

Figure 2.24: The regions of optimal solutions based on separability do not overlap with those based on the new capacity. **(a)** Separability and capacity landscape for a 4-cell TI code with a fixed first frequency: $f_1 = 1$. **(b)** Less than half of the good solutions based on capacity overlap with that based on separability in the non-shaded region of (a).

Loss 1 map with the new capacity for 4-cell and 6-cell codes. In Figure 2.25, I plotted the Loss 1 map for 4-cell and 6-cell codes for a comparison with the Loss 1 map in Figure 2.12. The new Loss 1 is simply defined as the negative capacity. In these maps, locations of local maxima (bad solutions) are of rational frequency ratio (same as before). A major difference is that the minima (good solutions) this time are local islands separated by large energy barriers. This implies that a good solution should have very specific coding line structure, and a perturbation on its frequency will render the solution no longer high-capacity. This result is in a drastic contrast to the Loss 1 map in Figure 2.12 where the good solution regions connect and form continents.

While this new capacity measure seems promising for the emergence of grid cell codes, there is an immediate issue needed to be addressed—i.e., the capacity is not differentiable with respect to optimization parameters. Next, I will address this issue

by proposing a variation in the current capacity measure.

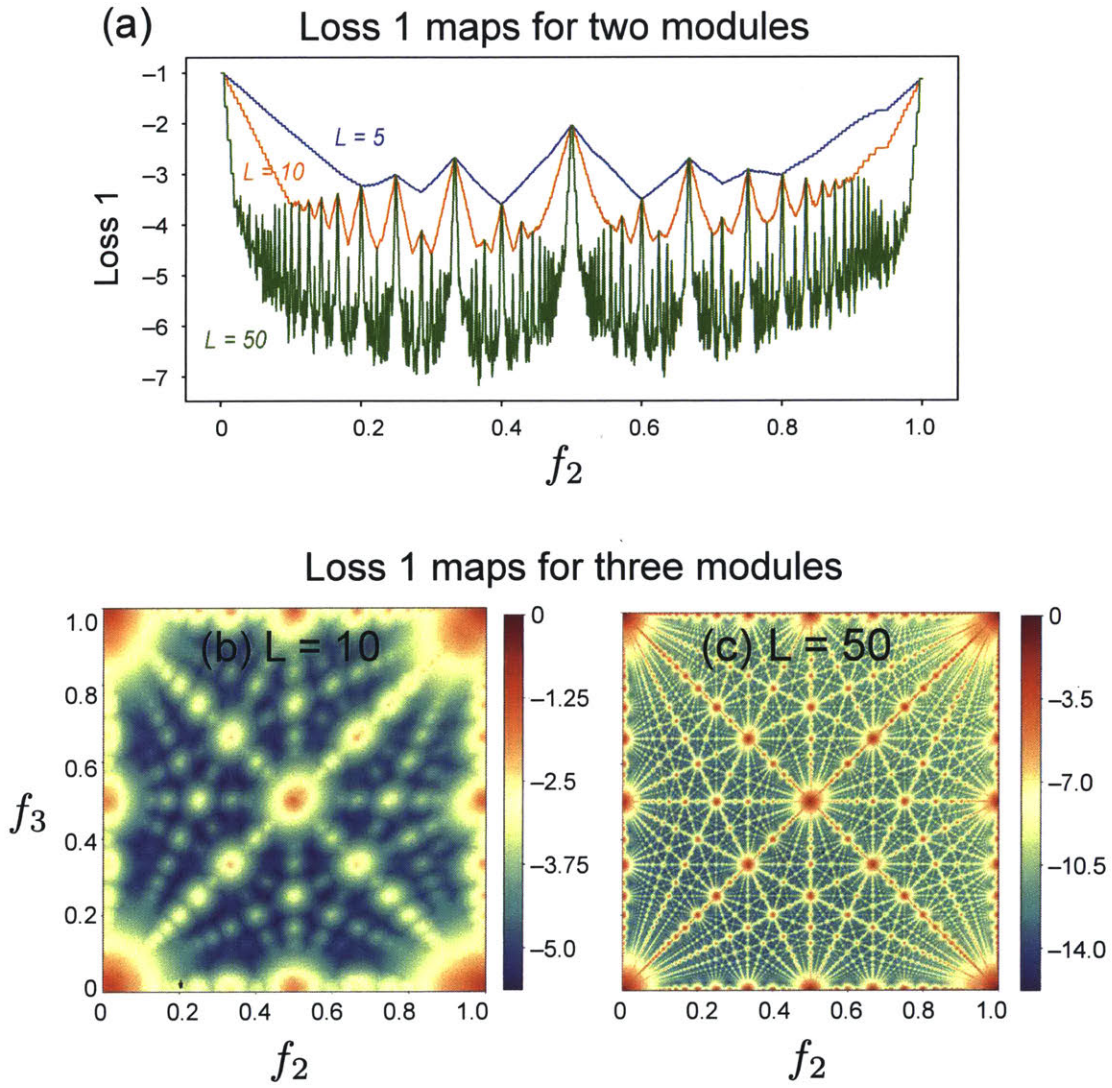

Figure 2.25: A maps of energy—negative capacity as Loss 1—landscape for 4- or 6-cell TI codes. A good solution is surrounded by high-energy barriers in isolation. **(a)** Landscape of a 4-cell TI code with fixed frequency for first cell pair $f_1 = 1$. **(b,c)** Landscape of a 6-cell TI code with coding range $L = 10w$ and $50w$ respectively.

2.6.2 How modules might emerge with the new capacity measure?

The intuition from Section 2.5.3 about how modules might emerge was wrong. Fundamentally, a capacity measure like average Euclidean distance—as a smooth function of the tuning curves—cannot give rise to an energy landscape in solution space such that the modular solutions are island minima. This is important because modular

solutions must form islands for them to be optima (See a simple topological argument in Appendix A.6 which shows that any two high-capacity modular solutions are necessarily separated by low-capacity non-modular solutions.)

Given the fact that the new capacity measure is a piecewise discrete function of tuning curves (See Figure 2.23(f) or Appendix A.7 for more details), it is very likely that the energy landscape shares similar island structure (Figure 2.25 as an example) even in high-dimensional space. For this reason, optimizing the new capacity measure is likely to give rise to modular solutions like grid cell codes.

2.6.3 Differentiable capacity measure with softmax

The reason for the current capacity measure indifferentiable is because the $\min(\cdot)$ operation in its definition. To remove this problem, I use a heuristic softmax function that properly weight the probability of a collision point at any positions, and thus estimate where this collision point will be.

New capacity measure.

$$C(d_{thres}) \equiv \min_{|x-x'|} \arg[d_{xx'} < d_{thres} \mid (|x-x'| > \Delta)], \tag{2.101}$$

Soft version of new capacity measure.

$$\text{softC}(d_{thres}) \equiv \int_0^L \int_0^L |x-x'| \cdot P(x, x' |\ d_{thres}, \rho, \Delta)\ dx\ dx' \tag{2.102}$$

where d_{thres} sets the strength of threshold noise, and

$$P(x, x' \mid d_{thres}, \rho, \Delta) = \begin{cases} \text{softmax}(x, x' \mid \rho, d_{thres}) & \text{if } |x-x'| \geqslant \Delta \\ 0 & \text{if } |x-x'| < \Delta \end{cases} \tag{2.103}$$

is a well normalized probability function of the capacity being at $x > x_0 \equiv 1/2$. A heuristic softmax function is given as

$$\text{softmax}(x, x' \mid \rho, d_{thres}) \equiv \exp\left[\rho \cdot \left(-\frac{d_{xx'}}{2\ d_{thres}} - \frac{|x-x'|}{(2 - d_{thres} + \epsilon)\ L}\right)\right], \tag{2.104}$$

where ρ sets the "stiffness" of the softmax function, $\epsilon = .001$ is a small number to prevent a vanishing numerator. The first term in the exponent amplifies probability

of certain displacements $|x - x'|$ with small distances $d_{xx'}$ compared to threshold noise d_{thres}, i.e. $d_{xx'} < d_{thres}$. The second term biases the these small distances toward the ones with smallest displacement: $|x - x'| \to 0$, i.e., $\min_{|x-x'|} \arg(d_{xx'})$. Note that for a strong threshold noise: $d_{thres} \to 2$, the probability of a $d_{xx'}$ where $|x - x'| = 0$ is always the largest.

To test how close this soft version of capacity resembles the original, I computed 6 examples in Figure 2.26 ranging evenly from $f_2/f_1 = 2/\pi$ to $f_2/f_1 = 1/\varphi$. As one can see the soft capacity function follows the original very well in various threshold noise.

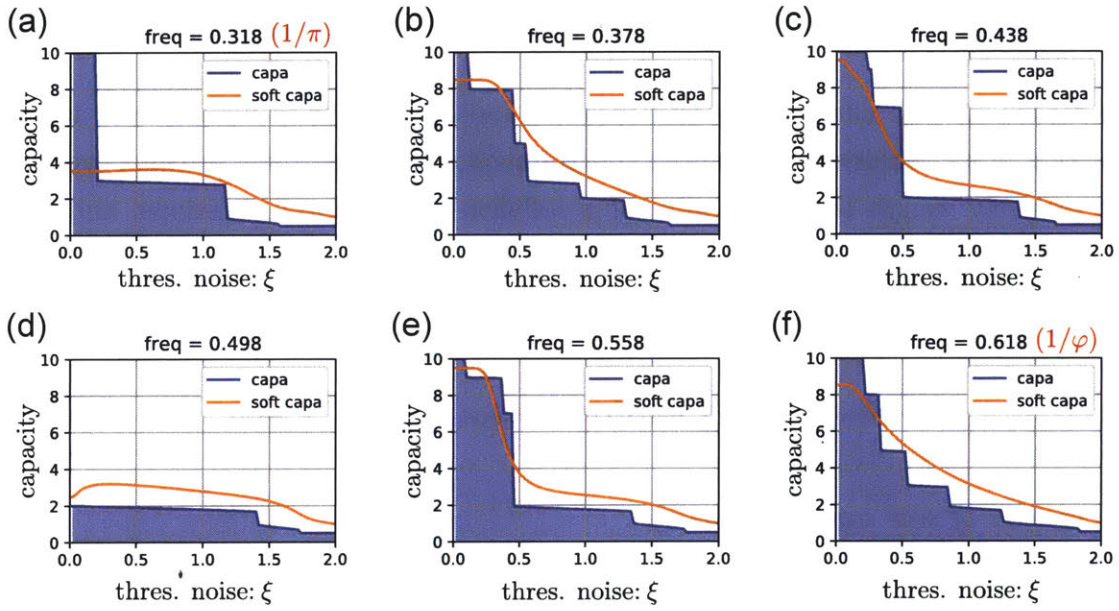

Figure 2.26: Soft version of the new capacity is differentiable and ready for optimization in Scheme C. (a-f) six examples of different 4-cell TI codes show a good match between the soft capacity and the original.

2.6.4 Issues to overcome

This last simulation is meant to be a preliminary run for pointing out the issues regarding approaching global optimum. The setup of Scheme C is identical to that of Scheme B except Loss 1 now implements the new capacity measure. There was only a minor fine-tuning on the relative firing rates (Appendix A.14).

Setup of Scheme C

I. Simple scheme without denoiser.

$$z_i(x) = \sum_{j=1}^{N_b} P_{ij}\ b_j(x), \tag{2.105}$$

where $i = 1, ..., N$, $x \in [0, L]$, and $b_j(x) \equiv \cos(2\pi f_{u_j} x + \phi_{v_j})$

with $\begin{cases} u_j & = 1, ..., U \\ v_j & = 1, ..., V \\ j & = 1, ..., N_b \equiv UV \end{cases}$. The frequencies and phases are equally divided: $f_{u_j} \in [0, 1/w]$ and $\phi_{v_j} \in [-\pi, \pi]$.

II. Soft version of new capacity measure.

$$\text{softC}(d_{thres}) \equiv \int_0^L \int_0^L |x - x'| \cdot P(x, x' |\ d_{thres}, \rho, \Delta)\ dx\ dx' \tag{2.106}$$

where d_{thres} sets the strength of threshold noise, and

$$P(x, x' \mid d_{thres}, \rho, \Delta) = \begin{cases} \text{softmax}(x, x' \mid \rho, d_{thres}) & \text{if } |x - x'| \geqslant \Delta \\ 0 & \text{if } |x - x'| < \Delta \end{cases} \tag{2.107}$$

is a well normalized probability function of the capacity being at $x > x_0 \equiv 1/2$. The heuristic softmax function is given as

$$\text{softmax}(x, x' \mid \rho, d_{thres}) \equiv \exp\left[\rho \cdot \left(-\frac{d_{xx'}}{2\ d_{thres}} - \frac{|x - x'|}{(2 - d_{thres} + \epsilon)\ L}\right)\right], \tag{2.108}$$

where ρ sets the "stiffness" of the softmax function, $\epsilon = .001$ is a small number to prevent a vanishing numerator.

III. Loss functions. All loss functions remain the same as those of Scheme B-2 except Loss 1 is now to maximize the new capacity.

$$L1 \equiv -\text{softC}(d_{thres}) \tag{2.109}$$

The second loss function takes a softer form.

$$L2 = \log \frac{1}{L^2} \sum_{x,x'} \left(D^2(x, x') - D^2_{tar}(x, x')\right)^2, \tag{2.110}$$

where $D^2_{tar}(x,x')$ replaces the value of each entry in $D^2(x,x')$ to the mean along its diagonal:

$$D^2_{tar}(x,x') = \frac{1}{L-\Delta}\sum_{x=1}^{L-\Delta} D(x,x+\Delta) \tag{2.111}$$

The constraint on demanding nonnegative tuning with a moving target bounded to the final target:

$$L3 = \log\ \left(r_z^2 - \theta\left(.99\ r_z^2, r^2_{z,tar}\right)\right)^2, \tag{2.112}$$

where $r_z^2 \equiv \frac{1}{NL}\frac{\sum_{i,x} z^2_{-,i}(x)}{\sum_{i,x} z^2_{+,i}(x)}$ with $\begin{cases} z_{-,i}(x) & = [z_i(x)]_- \\ z_{+,i}(x) & = [z_i(x)]_+ \end{cases}$ is the negativity of tuning curves $\mathbf{z}(x)$,

$\theta(u,u_{tar}) = \begin{cases} u & \text{if } u > u_{tar} \\ u_{tar} & \text{if } u \leqslant u_{tar} \end{cases}$ is a clip function, and

the target $r^2_{z,tar}$ is a chosen and fixed number.

IV. Soft constraint on codeword length in stochastic orthogonal gradient descent.

$$L0 = \sum_x \left(1 - \frac{1}{N}\sum_i z_i^2(x)\right)^2 \tag{2.113}$$

V. Multiple objectives. Stochastic orthogonal gradient descent is used on three main loss functions.

$$\min_P\ [L1,\ L2,\ L3], \tag{2.114}$$

VI. Constraint on population firing rate. The constraint on total firing rate remains the same:

$$\sum_{i,x} z_i^2(x) = L \tag{2.115}$$

Results of Scheme C are shown in Figure 2.27 The issue is immediately clear in Figure 2.27(a) where the 100 optimized solutions do not give rise to a visible potential Pareto front, indicating that these solutions are trapped in bad local minima and thus far from optimal. This issue is anticipated from the Loss 1 map of 4- and 6-cell codes; that is, the good solutions are islands separated by large energy barriers. Even though the introduction of the soft capacity enabled a gradient-based optimization method. It didn't fundamentally change the structure of energy landscape. And an optimization method like stochastic gradient descent will still have a hard time to approach the global optimum.

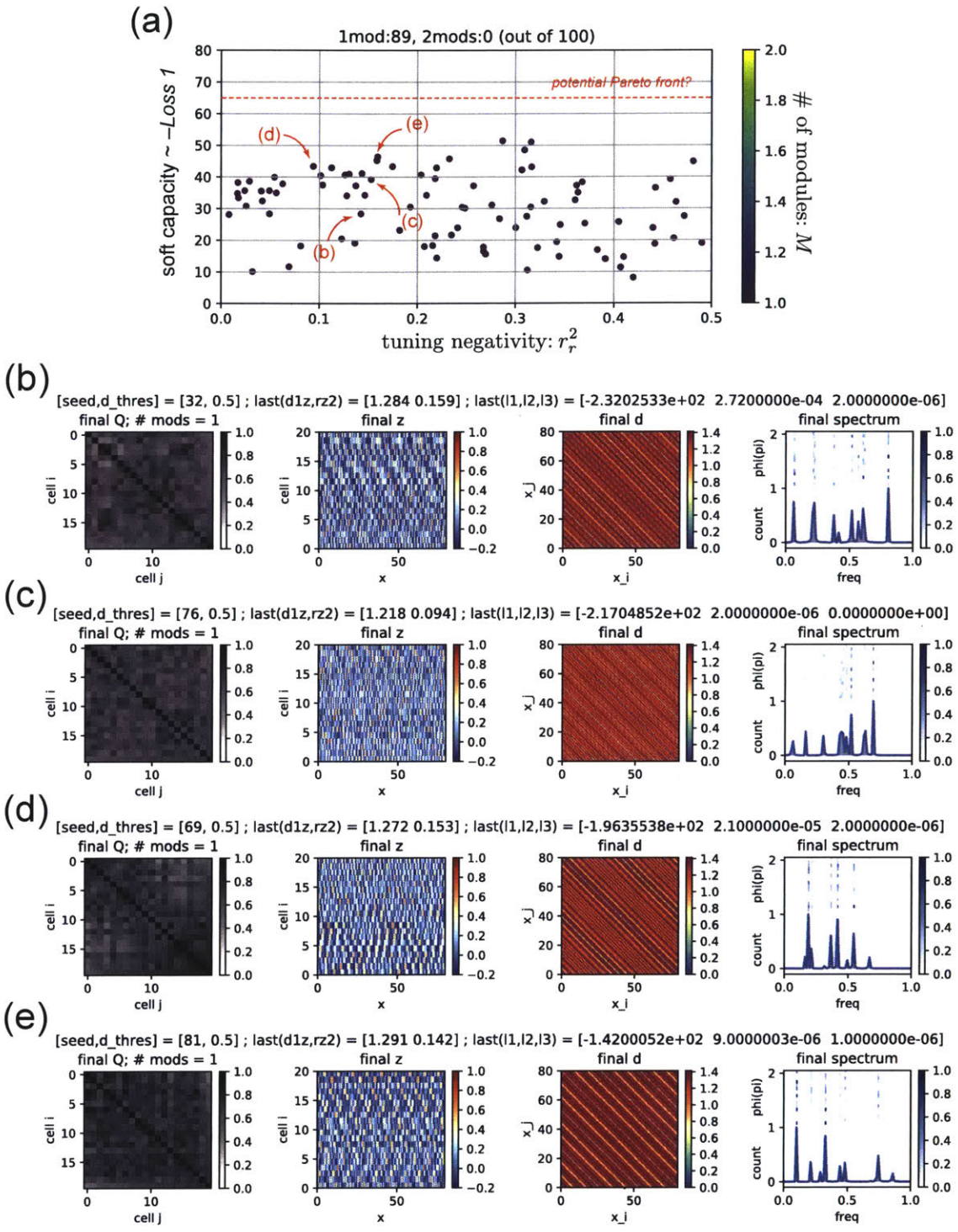

Figure 2.27: Optimized solutions are yet far from the global optimum. **(a)** The fact that there is no visible Pareto front implies that these optimized solutions are far from the global optimum. **(b-e)** Four example solutions in (a). These solutions do not resemble grid cells. They are all of single-module based on similarity matrices Q (See Appendix A.3 for the definition) and have non-periodic tuning curves.

2.6.5 Outlook and remarks of Scheme C

I have a couple of general suggestions to the issues raised above following the wisdom from either physics or, more recently, deep learning. The detailed implementation of these suggestions will still require judicious strategies and experimentation.

First, to follow the conventional wisdom from theoretical physics about studying minimal models, one may try to narrow down the search region to a few general solution types. For example, parameterizing tuning curves differently by forcing them into couple of modules. One can fix the number of modules for each set of runs and later compare among groups of optimized solutions with different number of modules. Same can also apply to parameterizing types of non-modular solutions and comparing their performance to that of modular solutions. This approach of re-parameterization not only narrows down the search region in the original full solution space, but potentially re-sculptures the optimization problem for avoiding island minima from a rough energy landscape. Although the approach excludes lots of the other possible solutions, it is nevertheless a good starting point for approaching global optimum.

Having said that, it is entirely possible that, even within a narrower solution type, the feasible parameterization of tuning curves still give rise to a rough energy landscape. For this, one can attempt to map out entire energy landscape by carefully sampling the initial conditions. The mapping in parameter space become more feasible because the search region has been narrowed down drastically after re-parameterization. To efficiently map out the energy landscape, one can hierarchically shrink the mapping regions. E.g., first running 100 initial conditions that coarsely cover full parameter range, and secondly focusing on the search region near the top 10% of good solutions. Doing this 100 runs hierarchically each time with an ever more narrowed search region should improve the percentage of good solutions, and eventually the last 100 runs should be closer to global optimum for all solutions.

Alternatively, one may follow the wisdom from deep learning and operate in an over-parameterization regime. A deep neural net can be a suitable candidate function approximator for our purpose given its latest successes (Bengio, 2012; Bottou, 2012). The rationale behind using a multilayer feedforward network for effectively constructing a desired function is to have many more parameter configurations—in close vicinity with each others—that give rise to a single tuning curve solution. If so, these desired solutions could form continent in configuration space and might become largely accessible by gradient-based optimization approaches.

2.7 Chapter summary

- The descriptive function hypothesis I adopted focuses on two coding properties of grid cells: *Grid cells exist for having a high-capacity and robust path-integrating code.*

- This chapter focuses on setting up various optimization schemes and quantitative measures to test this function hypothesis. These optimization schemes can be broadly categorized as 1) RNN training approach and 2) coding theoretic approach.

- The unsuccessful demonstrations of the emergence—from both mine and Deepmind's RNN training schemes—may provide insights to the existence of the *function complementarity* in place-grid cell dual system in the brain. Specifically, it supports two conjectures about grid cells: 1) a grid cell circuit is evolutionarily hard-wired through a non-Hebbian-type plasticity mechanism during development; 2) a grid cell circuit—to a large degree—cannot be modified via synaptic plasticity without compromising its high-capacity encoding.

- A coding theoretic approach removes RNN dynamics and only focuses on formulating quantitative measures as functions of tuning curves themselves.

- A coding theoretic approach is legitimate for pursuing an optimization principle in the context of grid cells, because RNN learning dynamics (learning a circuit) is irrelevant to the function hypothesis.

- A binary grid cell code optimizes its coding range (capacity) under the constraint of being TI—short for translationally invariant (associated to path-integration); whereas, a continuous grid cell code optimizes 1) coding capacity, 2) TI constraint, and 3) an additional biological constraint.

- Scheme A targets modular sinusoidal tuning curves. It requires an extra denoiser and a complex capacity measure for leveraging a multi-module—instead of a single-module—solution to emerge as the optimum. This implausible requirement of an denoiser and complex capacity measure is caused by the assumption on targeted tuning curves being *sinusoidal.*

- Scheme B targets modular, periodic, narrow, positive tuning curves—with a shape of von Mises function. It doesn't require a denoiser and has a capacity

measure as simple as an average Euclidean distance. The optimized solutions had one or two modules and clustered towards a potential Pareto front indicating global optimality. The double-module solutions—some of them resemble grid-like tuning—are minority and less in capacity than single-module solutions.

- Scheme C reformulates the capacity measure—targeting a robust long coding line without critical failure. A soft version of new capacity measure was proposed for circumventing the issue of indifferentiability. The energy landscape of the new capacity measure is rough such that a local minimum is surrounded by high-energy barriers in isolation. A test run using modified Scheme B showed that optimized solutions are yet far from the global optimum. The major challenge for future studies is on means to approach the global optimum.

Chapter 3

Place cells emerge as an optimal learnable representation guided by grid cells

3.1 Introduction

In this chapter, we will see how place cells possess coding and dynamical properties that are *complementary* to grid cells. Unlike grid cells, a place cell code is changeable through experiences; for that, the focus of Chapter 3 is more about understanding the role of dynamical constraints in the optimization principle and less about studying static coding properties (which are the focus of Chapter 2). The main goal of this chapter is therefore to fulfill the core constraint, i.e., to finding a neural implementation as a learnable recurrent neural network (RNN) that can continuously learn new representations for novel environments without the issue of catastrophic forgetting caused by severe synaptic overrides.

3.1.1 Function hypothesis: Learning distinctive codewords while retaining the old

In comparison with grid cells, the dynamics of place cells is much richer and contextual for which it leads to a much larger mass of literature and function hypotheses. To keep the focus, I will limit the discussion only to the low- or high-level functions that directly relate to the aim of this chapter—i.e., finding a plausible neural imple-

mentation of a rapidly learnable spatial code. It is necessary to mention that CA3 is the focus of this thesis out of all the other subregions in the hippocampus because of its rich recurrent connectivity that potentially enables a standalone path-integrator, a requirement for the *function complementarity* introduced in Chapter 1. Although CA3 cells were also hypothesized to play a functional role in pattern separation and completion in nonspatial memory tasks, I will narrow the discussion within its functions related to spatial tasks—hence to respect the name: *place cells.*

Low-level functions of CA3 place cells

Below, I relist the tuning properties of place cells introduced in Chapter 1 with hints of their corresponding coding properties.

1. *High-spatial resolution*: They have narrow unimodal spatial tuning curves in a relatively small environment (Maurer et al., 2005).

2. *Uniform spatial coverage*: Their tuning curves uniformly cover the space of an environment (Muller et al., 1987).

3. *Global remapping*[1]: They have near orthogonal set of tuning curves such that the cofiring structure of a certain environment is uncorrelated to the other ones (Muller et al., 1987; Alme et al., 2014).

4. *Learnable representation*: A new set of tuning curves can be rapidly learnt when encountering a novel environment (Wilson and McNaughton, 1993; Frank et al., 2004).

5. *Biased field propensity*: Some place cells always have more fields than the others in either a small or a large environment (Rich et al., 2014; Lee et al., 2019).

From these tuning properties, one can hypothesize their corresponding coding properties as low-level functions:

1. *Being a path-integrating code*: In parallel to the grid cell uniform phase coverage within a module, the uniform spatial coverage of place cell tuning curves

[1]Unlike grid cells which largely preserve their tuning curves (up to a rotation and translation) and cofiring structure (Fyhn et al., 2007; Leutgeb et al., 2007), place cells randomize their cofiring structure in a different environment. This decorrelated tuning property across environments—as much of an independent function as it appears to be—can simply be a direct consequence of Function 1 if one takes a parsimonious interpretation.

could be for path-integration. However, note the subtle difference for which the translational invariant (TI) coding property is not explicitly needed in case of place cells for a reason to be discussed in Section 3.1.2

2. *Having high coding separability*: A direct consequence from having 1) a narrow unimodal tuning curve for each cell and 2) a uniform spatial coverage for an ensemble is to have a spatial code that is maximally orthogonal (a result discussed in Section 2.5.2). In other words, any two place cell codewords are highly separable.

   There were a lot of speculations about the potential functions for global remapping. Here I take a parsimonious interpretation: A global remapping is a direct consequence of seeking a new segment of place cell code that is maximally orthogonal to the existing code. In that regard, a global remapping is within the function of having *high coding separability.*

3. *Having a learnable representation*: This function is better classified as a dynamical property than a coding property for reasons discussed in Section 3.1.2. In short, because new place fields rapidly form in a novel environment. The time it takes for an algorithm to search for orthogonal codewords needs to be constrained.

4. *Functions of a biased field propensity*: It has been hypothesized (Lee et al., 2019) that such a bias is to perform both a rapid recognition of an environment (task of classification) and to have a high spatial resolution within an environment (task of self-localization). The idea is that a cell that rarely fires provide straightforward information about which environment the rat is in (because it only fires in very few environments); whereas a cell that fires all the time provides necessary spatial resolution for having a path-integration capability.

Following the guidance of having minimal assumptions, the descriptive function hypothesis I provide below is based on the first three functions. The biased field propensity, as it turns out, can emerge in an RNN already based on just the first three functions without optimizing the fourth. The function hypothesis for place cells thereby states:

> *Place cells exist for having a sequentially-learnable and highly-separable path-integrating code.*

High-level functions of place cells

Without restricting to the CA3 place cells, I listed few relevant high-level functions that have been hypothesized for flexible behaviors in a spatial task.

1. *Topological codes*: Place cells code is hypothesized to be topological (Curto and Itskov, 2008; Curto, 2016; Curto et al., 2017; Dabaghian et al., 2012, 2014; Babichev and Dabaghian, 2018; Low et al., 2018; Chen et al., 2012, 2014) such that it mainly captures underlying graph structure of a task without excessively devoting resources for a precise metric encoding as what an ensemble of grid cells does.

2. *Predictive maps*: From asymmetric or reward-biased tuning curves observed in experiments (Hollup et al., 2001; Zheng and Colgin, 2018), place cells has been hypothesized as successor representations in a reinforcement learning (RL) task (Stachenfeld et al., 2017; Gershman, 2018; Gardner et al., 2018) that contain information about transition probability for a learnt policy which can lead to an optimal cumulative reward.

In Section 3.2.4, I will discuss how topological code and predictive maps can be a higher-order learning as an extension of the first-order learning based on hypothesized low-level functions.

3.1.2 Optimization principle: Focus on finding learnable neural implementation

Given that the hypothesized functions for place cells are complementary to grid cells, the approach for pursuing an optimization principle also has to be different. Because place cells can learn sequentially in such a way that RNN learning dynamics is an essential part of the optimization problem, a pure coding theoretic approach—introduced in Chapter 2 for grid cells—is no longer applicable. Having said that, we may still apply a coding theoretic approach along with an RNN training approach because each of the two hypothesized functions for place cells can be described as nicely as either a coding or a dynamical property:

> Function 1 is a coding property: *Having a spatial code with high separability within and across environments.*

Function 2 is a dynamical property: *Capable of learning a novel environment with ability to recall a learnt one.*

To optimize Function 1, one may consider a coding theoretic optimization scheme introduced in Chapter 2, i.e., Scheme B-1 in Section 2.5.2, in which both the *separability* (average Euclidean distance between all pairs of codewords) and the *nonnegativity* (which demands positive tuning curves) are optimized simultaneously using the *stochastic orthogonal gradient descent* (See Appendix Appendix A.5 for details). From the results replotted below, in Figure 3.1(a), one can see that an optimal solution given enough cells always has 1) the highest separability that is theoretically bounded and 2) unimodal tuning curves with the narrowest achievable width. Note that the optimal solution is both locally and globally TI even though such a constraint is not explicitly applied in the optimization scheme. The local TI property is achieved implicitly from the fact that all tuning curves have narrowest allowable width and they tile the space evenly (a condition allows a simple mechanism for path-integration as discussed in Section 2.3.3). The global TI is achieved because all distant pair of codewords are orthogonal to each other.

The tuning curves from the above optimal solutions agree with the static tuning properties of place cells. The other half concerns the *dynamic* tuning property—i.e., Function 2. To reconcile Function 1 and Function 2, which demands sequential learnability, one may consider Function 2 as an algorithmic constraint for optimizing Function 1. In other words, the process of optimizing Function 1 needs to be *online*, rendering Scheme B-1 an unsuitable approach for the optimization principle. In Section 3.2, I will introduce an *optimal online learning algorithm* that aims to approach the optimal solutions in Scheme B-1 under the constraint to rapidly construct tuning curves as the agent moves. This optimal algorithm can be seen as a preparation to fulfill the final core constraint in the optimization principle: a *learnable* neural implementation, as will be introduced in Section 3.5.

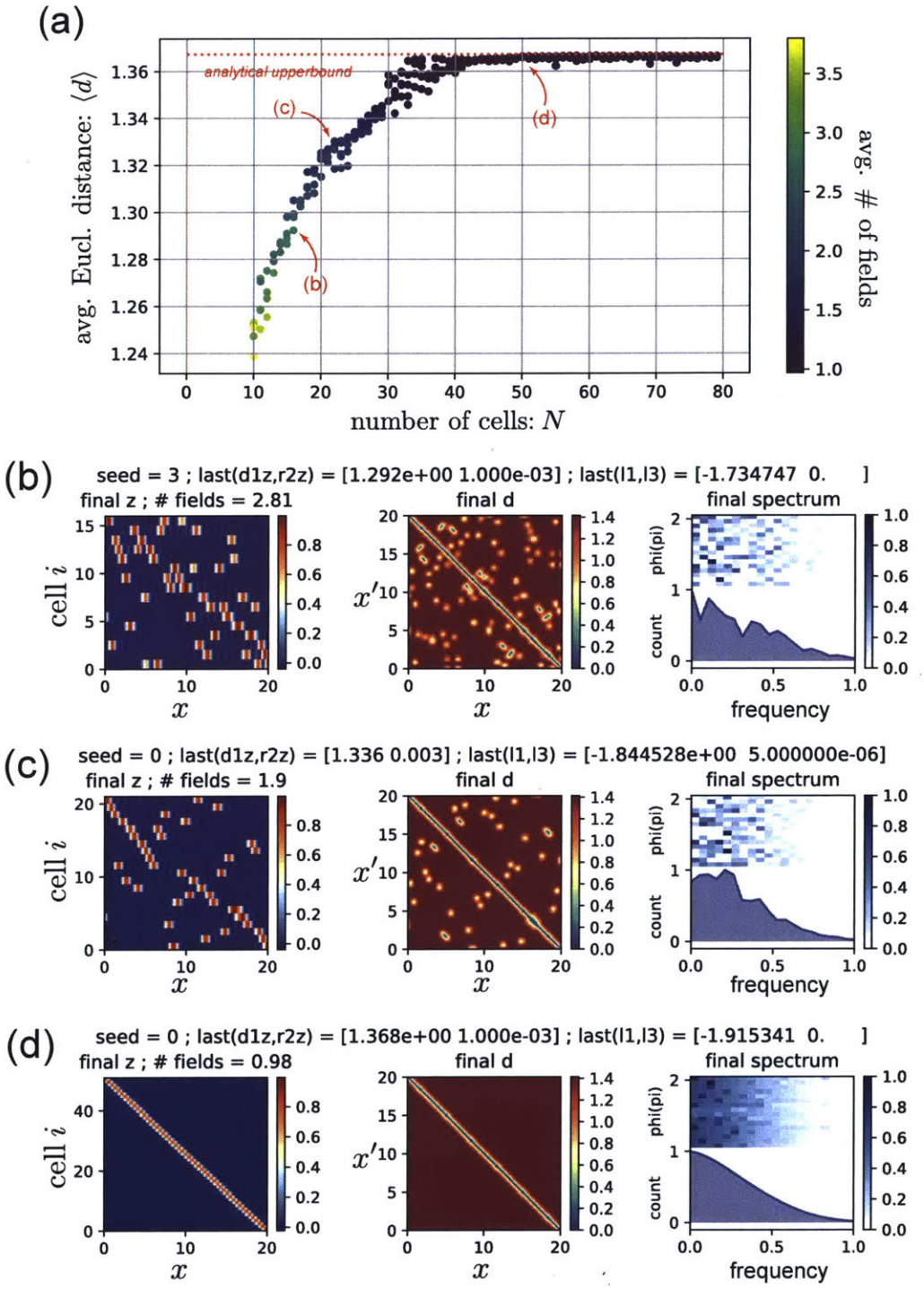

Figure 3.1: An optimal solution is unimodal given enough cells in Scheme B-2. **(a)** With a fixed maximal frequency $1/w$ in Fourier basis functions, an optimal solution has one field per cell if the number of cells, N, satisfies $N > L/\sigma = 20/.48 = 42$. (b-d) Three examples with different N.

3.1.3 Chapter organisation

The rest of Chapter 3 has three parts. Part 1, or Section 3.2, focuses on finding an optimal online learning algorithm that give rise to place cells within the framework of coding-theoretic approach. Part 2, or Sections 3.3 and 3.4, focuses on overcoming the catastrophic forgetting—a well known problem in training a neural network. And the last part, Section 3.5, focuses on finding a plausible neural implementation of the optimal online learning algorithm based on the knowledge developed in Part 2 about how to circumvent the catastrophic forgetting. In both Part 2 and 3, I will discuss experimental implications and predictions mostly based on the inevitable dynamical constraints of an RNN which has been overlooked by the field in general.

3.2 Optimal online learning algorithm

In this section, I will introduce to an *optimal online learning algorithm*—that maximizes codeword separability (Function 1)—as a necessary step towards fulfilling the core constraint in the optimization principles. The algorithm is purely coding theoretic yet already capturing the online aspect of Function 2—*capable of learning a novel environment*—such that the algorithm can be directly adapted for a plausible neural implementation as will be discussed in section 3.5.

To keep the online algorithm simple, it needs a guiding source from a translational invariant (TI) spatial code that assists the formation of place fields. The most suitable candidate is no other than a grid cell code—to which I introduced in Section 1.1.3 as a part of the fundamental stance of this thesis: the *grid-assist architecture*. I will emphasize the minimalism of this architecture by comparing to using another high-level visual cue from object manifolds, e.g., from inferior temporal cortex (IT) in macaques, as a guiding source. The algorithm consists of two interleaving algorithmic steps: 1) optimal random landmark sampling and 2) online similarity matching. Both of them require the assistance from grid cells. But, importantly, after an RNN—as will be introduced in section 3.5—has learnt to generate a place cell code, it is expected to operate alone.

In the last two subsections, I will revisit the *function complementarity* (introduced in Chapter 1 and discussed in Section 2.2.4) from the perspective of place cells, and make a straightforward extension on the original online algorithm to incorporate the topological coding aspect of place cell code that has been observed in experiments.

3.2.1 Grid cell activity as an ideal cue to guide formation of new place fields

The main stance—the *grid-place function complementary*—is to have both the grid and place cell systems function largely independently. To achieve this, place cells cannot be a simple decoder of the grid cell code. Being a spatial code, however, a place cell code needs to acquire a similar neighbor relationship, like grid cells, among codewords in such a way that a place cell code is 1) locally translationally invariant and 2) orthogonalizing codewords beyond the close neighbors[2]. A spatial code like this is fundamentally allocentric (Leutgeb et al., 2013) such that it takes an allocentric guidance for the formation of the code. Grid cell code is obviously a suitable choice as a guide, but let us first discuss another possibility about what it takes to use the visual cue from a high-level sensory system as guidance.

Imagine an environment like Figure 3.2(a) on which there are three objects along the way. The stream of raw visual stimuli from these objects are represented and re-represented along a hierarchical feedforward pathway—e.g., the primate ventral visual stream: $Retina \longrightarrow RGC \longrightarrow LGN \longrightarrow V1 \longrightarrow V2 \longrightarrow V4 \longrightarrow IT$. This pathway, level by level, transforms a pixel image on the retina to a compressed representation that is most useful for solving relevant behavioral tasks. This representation at the top level, i.e. IT, has been hypothesized to form object manifolds (DiCarlo and Cox, 2007; DiCarlo et al., 2012; Chen et al., 2018; Chung et al., 2018; Cohen et al., 2019) such that it can solve classification problem easily with a biologically plausible decoder whilst keeping the encoding of various viewpoints for each object. The key is to store *flattened*[3] object manifolds and arrange them in parallel, so that they are *linearly separable*, and hence a classification task can be trivially solved.

Figure 3.2(b) illustrates how the state trajectories of an IT cell ensemble might go as the animal traverses in Figure 3.2(a). As one can see that in order to capture various visual aspects of an object (e.g. position, pose, size, and shape, etc.) the manifold needs to span into a few dimensions. As a result, the trajectory on a manifold highly depends on where and how this manifold is placed in the state space. And a flat manifold with linearly interpolating states does not provide translationally invariant cues either. Moreover, the activation of a manifold state depends on whether

[2]The property of orthogonalizing codewords beyond neighbors directly corresponds to maximizing codeword separability.

[3]Any points on a flat manifold can be expressed as a linear combination of other points on the same manifold.

the corresponding object is currently within the animal's view. Consequently, the state trajectories for proximal objects, though provide more position information, are short-lived; whereas, a long-lived state trajectory from a distal object provides much less position information (a distal object does not change shape or size much during traversing). In other words, even with a highly compressed representation at the top visual hierarchy, the visual cues are still viewpoint dependent. It is therefore unsuitable for directly assisting place field formation.

In contrast, a grid cell code is allocentric—a property directly inherited from being a translationally invariant spatial code. Figure 3.2(c) illustrates the state trajectory of a 1D grid cell code; one can see that the local structure is identical everywhere on of the coding line, a feature inherited from being a TI code. In other words, a grid cell code is not contextual and contains purely location information, and thereby an ideal cue to guide formation of new place fields.

3.2.2 Optimal learning as online sparse manifold transformation

Unlike a grid cell code, a place cell code is only locally translationally invariant. This crucial difference makes it possible to construct a place cell code online since the correlation between two distant codewords is not a quantity directly relevant to be optimized. Having Known that, the algorithm, also named the online sparse manifold transformation, can be broken down into two interleaving steps as explained in Figure 3.3.

Step 1: Optimal random landmark sampling

The main idea is to firstly lay down orthogonal codewords each encodes a sampled landmark location that tiles the entire environment, and secondly do local interpolations between any adjacent locations for establishing neighboring relationships among these codewords. To do this rapidly while animal is running, the algorithm is not allowed to compare a currently sampled codeword to the previous ones. It is therefore no guarantee for a new sample codeword to be orthogonal to all previous sampled codewords. The best one can do under such a constraint is to sample a sparse codeword with each cell being drawn at equal probability. Ideally, the more sparse a sampled codeword is, the less overlap among all randomly sampled codewords. Prac-

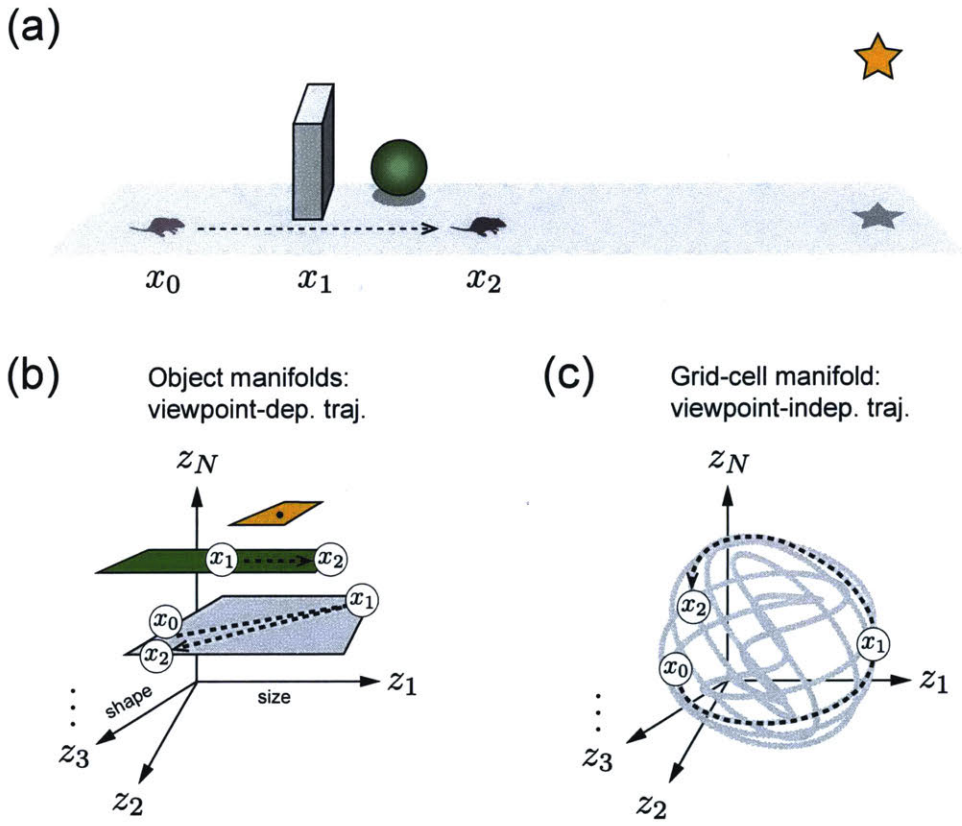

Figure 3.2: A grid cell code is an ideal allocentric cues for guiding a formation of new place fields. **(a)** A three-object environment that provides distinct visual scenes for the animal to identify its location. **(b)** Compressed representations known as object manifolds in higher visual system, e.g. IT in a macaque, organized as parallel and flat segments. The dash lines on the manifolds depict the state trajectories when the animal traverses along a straight line. These sequential activation on object manifolds are highly viewpoint-dependent (egocentric): e.g. the brick changes both size and shape when the animal moves while the ball changes only its size yet was blocked until the animal passed through x_1, meanwhile, the distant star do not change shape or size during the traversing. Note, however, that the shape or size of individual objects need not to be encoded in the same direction. The manifolds are drawn this way simply for the explanation purpose. **(c)** A grid cell manifold is allocentric such that the local progression is identical, or translationally invariant. It is thereby an ideal cue for forming an allocentric place cell code.

tically, one should use a finite sparsity for a sampled codeword to achieve desirable noise resistance. As an example, the CA3 place cell ensemble has the fraction of activation below 3% (with a rate threshold above 1.5 Hz) at any given time (Leutgeb et al., 2004).

As for the sampling rate, it is optimal to have the distance between two sampled landmarks equal to half the minimal allowable tuning width (See a discussion on the relation between noise and tuning width in Section 2.3.4). In such a case, the place

cell code locally reach the highest spatial resolution—or maximal separability. The minimal observed tuning width in rodent grid cells or place cells is about one body size $\sim 20cm$, so it makes a 10cm interlandmark distance.

As an example, assuming an optimal sampling process once every $10cm$ with 1% activation using a uniform distribution. In doing so, it will take 100 times random landmark sampling before a substantial overlap between any two sampled codewords—which translates to a coding range of $10m$ on a 1D track. In other words, an environment smaller than $10m$ is considered to be small and a place cell code under the online construction is almost as good as an offline construction (maximally separable) that ensures all codewords at sampled landmarks are nearly orthogonal.

Step 2: Online similarity matching

Once a new sampled codeword is assigned to a landmark location, the process of *online similarity matching* searches for a series of overlapping codewords to fill in the gaps between the current and the last sampled codewords. As illustrated in Figure 3.3(a), these codewords in the gap need to interpolate between the adjacent landmark codewords such that all the codewords within the gap have correlation structure that *locally* matches the correlation structure of the grid cell code—e.g., Figure 3.3(d,e). The process of similarity matching is *online* in the sense that only the correlations between neighboring codewords—not the distant pairs—are required to match that of grid cells. The memory required in the process is a time interval as short as 1s; the biological plausibility of this synaptic memory will be discussed in Section 3.5.2.

An example of the resultant place cell codes out of Step 1 and 2 will have a distance matrix looks like Figure 3.3(d); note that it matches grid cell distance matrix in Figure 3.3(e) only locally. The corresponding coding line is illustrated as Figure 3.3(b) in which all the codewords that encode sufficiently faraway locations are distant with each other in state space. On the other hand, this is not necessarily true for grid cells where two distant locations can be encoded with close by codewords as shown in Figure 3.3(c). It is worth to note that although the online similarity matching only interpolates the two adjacent landmark codewords locally, they are automatically orthogonal to all the distant landmark codewords or their corresponding interpolating codewords (See Appendix B.1 for details). Therefore a distance matrix like Figure 3.3(d) can be achieved even without explicitly imposing a global

orthogonality constraint.

***Online sparse manifold transformation* is not to decode grid cells, but to learn an independent spatial code**

It is important to stress that the online manifold transformation introduced here is not simply building a feedforward decoder to find a one-to-one mapping with preserved codeword correlation: a method mostly used to find a compressed representation in a stream of visual data (Sengupta et al., 2018). The manifold transformation in my setting is to produce a place cell code independent of its guide, the grid cell code, such that a learnt place cell code can still function even after turning off the grid cell inputs. Note that there is no sense of turning off the grid cell inputs in the two algorithmic steps of online sparse manifold transformation. The importance of this conceptual distinction will only become clear after we are introduced to the RNN implementation in Section 3.5—in which both the landmark sampling and similarity matching process is to learn the recurrent connections among place cells instead of feedforward projections from grid to place cells.

3.2.3 Function complementarity revisited

This is a good point to revisit the *function complementarity* discussed in Section 2.2.4 from the place cell perspective. The three conjectures, regarding the existence of grid-and-place-cell dual system in the brain, are:

1. *A grid cell circuit is evolutionarily hard-wired through a non-Hebbian-type plasticity mechanism during development.*
2. *A grid cell circuit—to a large degree—cannot be modified through experience via synaptic plasticity without compromising its essential function—high-capacity encoding.*
3. *A place cell circuit can only learn from experience via synaptic plasticity—with a compromise in its coding capacity.*

In Section 2.2.4, we discussed how dynamical constraint could be the reason for the origin of the dual system in the brain. Specifically how grid cells—in order to achieve high coding capacity—compromise their learnability (*Conjecture 1* and *2*). For place

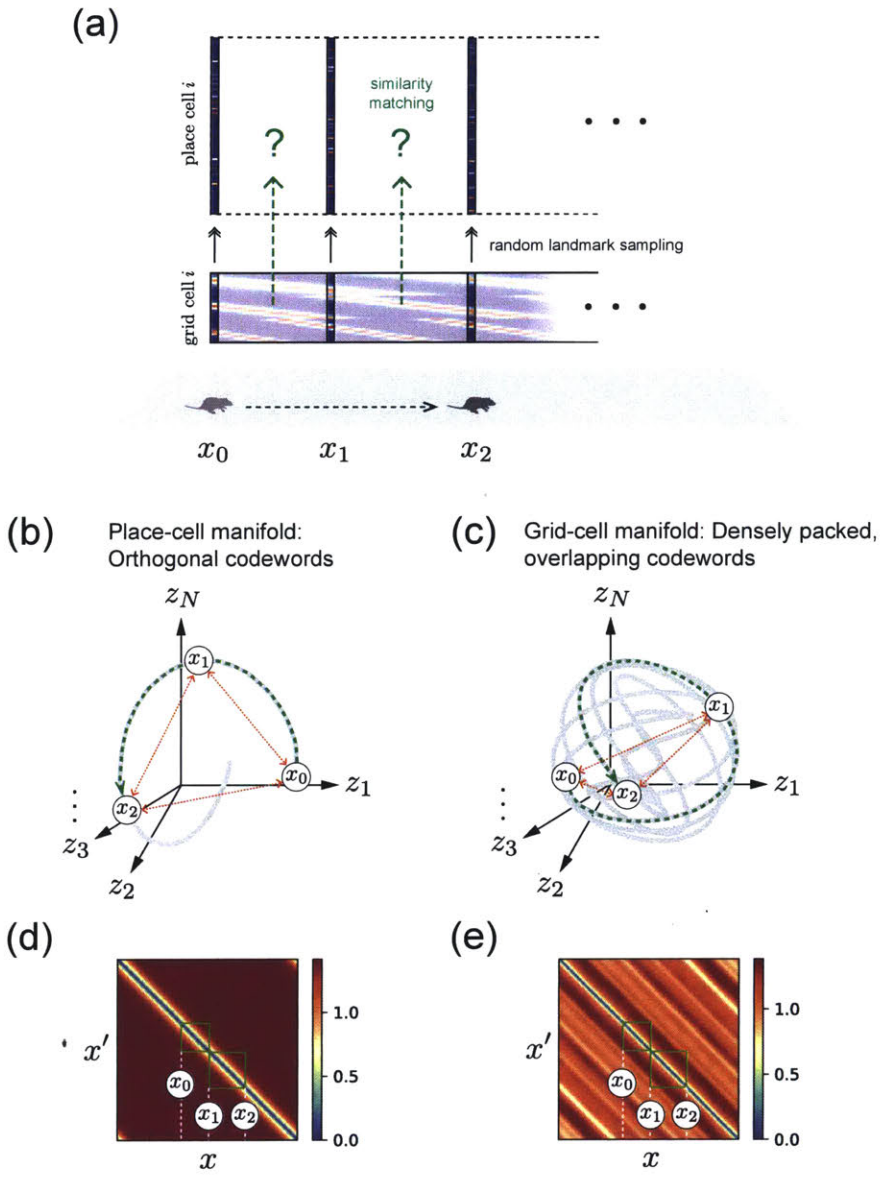

Figure 3.3: ***Online sparse manifold transformation*** **as an optimal algorithm that generates an independent place cell code with the assistance of grid cells.**. **(a)** The algorithm consists two interleaving steps: 1) the *optimal random landmark sampling* that sporadically generates codewords using the instantaneous grid cell input as a sampler, and 2) the *online similarity matching* that interpolates two adjacent landmark codewords. The sampled landmark codewords in Step 1) needs to be sparse and from a uniform distribution such that they are mostly orthogonal to each other. **(b,c)** A comparison between place and grid cell manifolds, in which a place cell manifold is shorter in length (or having a shorter coding range) such that a highly-separable code is affordable, whereas a grid cell manifold is lengthy and so requires the coding line to be nicely packed such that they have a descent separability above some threshold. **(d,e)** A comparison between place and grid codeword distance matrices. For a place cell code, all distant codeword pairs are maximally separable (orthogonal), whereas a distant codeword pairs for grid cells overlaps a lot.

cells (*Conjecture 3*), however, learnability is a major function such that their coding capacity is compromised. As one can see in the example given in Section 3.2.2, a code created from the online sparse manifold transformation has maximal separability only up to a coding range linearly proportional to the number of cells—assuming p active cells are required for any codewords. The fraction of active cells is p/N such that the coding range, before substantial overlaps among codewords, is

$$L < \frac{N}{p} w \propto N, \tag{3.1}$$

where w is the minimal tuning width. This result naturally comes about due to the online aspect of the optimal learning algorithm, which is the first requirement for having a learnable representation. In Section 3.5, we will discuss how the dynamical constraint of an RNN—on top of the constraint of being an online algorithm—further lower the coding capacity in Section 3.5.3.

3.2.4 Learning topological codes with flexible manifold transformation

It has been long hypothesized that a place cell code is fundamentally topological in the sense that it captures neighbor relations (e.g. A is next to B; B is next to C) among all encoded landmark locations instead of the precise distances between them (e.g. A is $20cm$ away from B; B is $30cm$ away from C). The topological aspects of a place cell code have been studied extensively both experimentally and theoretically (Curto and Itskov, 2008; Curto, 2016; Curto et al., 2017; Dabaghian et al., 2012, 2014; Babichev and Dabaghian, 2018; Low et al., 2018; Chen et al., 2012, 2014). Below, I provide two simple extensions to the *online sparse manifold transformation* to incorporate essential topological aspects of place cell codes.

I. Flexible random landmark sampling for building a topological code

From earlier discussion, we know that the place cell has a coding capacity approximately equal to the amount of orthogonal codewords that can be drawn from a random sampling. This capacity—measured in the number of orthogonal codewords—does not directly depend on the spatial scale. That is, the spatial coding range can either be small or large depending on the landmark sampling rate—which effectively changes the spatial resolution of a place cell code (Maurer et al., 2005).

As an example, Figure 3.4(a) shows that the landmark sampling rate is relatively higher around the middle region of the track where the sensory cues are the most rich. This reallocation of the available orthogonal codewords empowers a place cell code to cover a much larger spatial range. Because of this changeable interlandmark distances that underlies a topological code, the online learning algorithm requires an adaptable similarity matching assisted by a code with various spatial resolutions. Fortunately, the assisting grid cell code can again fulfill this role because it is modular with multiple scales (Stensola et al., 2012) and it can be compositional in such a way that a subset of few modules still forms grid cell code (still translationally invariant). As illustrated in Figure 3.4(c-e), one can therefore use modules with smaller grid spacings to guide the formation of place fields in the middle region, and those with larger grid spacings for place fields outside the middle region.

Alternatively, a similar augmented spatial code can also be achieved with a steady sampling rate but using a flexible sampling sparsity. Such a learning process could be suitable for encoding a small environment in which cells are relatively abundant and a high spatial resolution is affordable everywhere. In which case, a place cell code would dedicate more cells (lower sparsity) in encoding the middle region of the track for achieving higher noise resistance. In fact, this effect of a biased spatial coding has been observed in experiments (Hollup et al., 2001; Zheng and Colgin, 2018) in which the percentage of active place cells are higher near the reward location.

II. Learning trajectory-dependent spatial codes that underlie various graphs

From our discussion so far, it is not hard to draw a parallel between a place cell code and a graph. One might consider the sampled random codewords—that encode landmark location—as *vertices* and the interpolating codewords between them as *edges* on a graph. In which sense, one can extend the original *online sparse manifold transformation* to build a topological code that underlies a given graph. To illustrate how this can be done, I use four place cells to encode a square environment of nine discrete positions as an example—shown in Figure 3.5.

Although the agent was trained in the same environment, depending on which learning trajectory it learnt a distinct place cell code that underlies a unique graph. In Figure 3.5(a), the four corners can be considered as landmark locations and the other ones the interpolating codewords that connect them. Note that Codeword E only overlaps with Codewords 1 and 3 so that it can be seen as an edge connecting

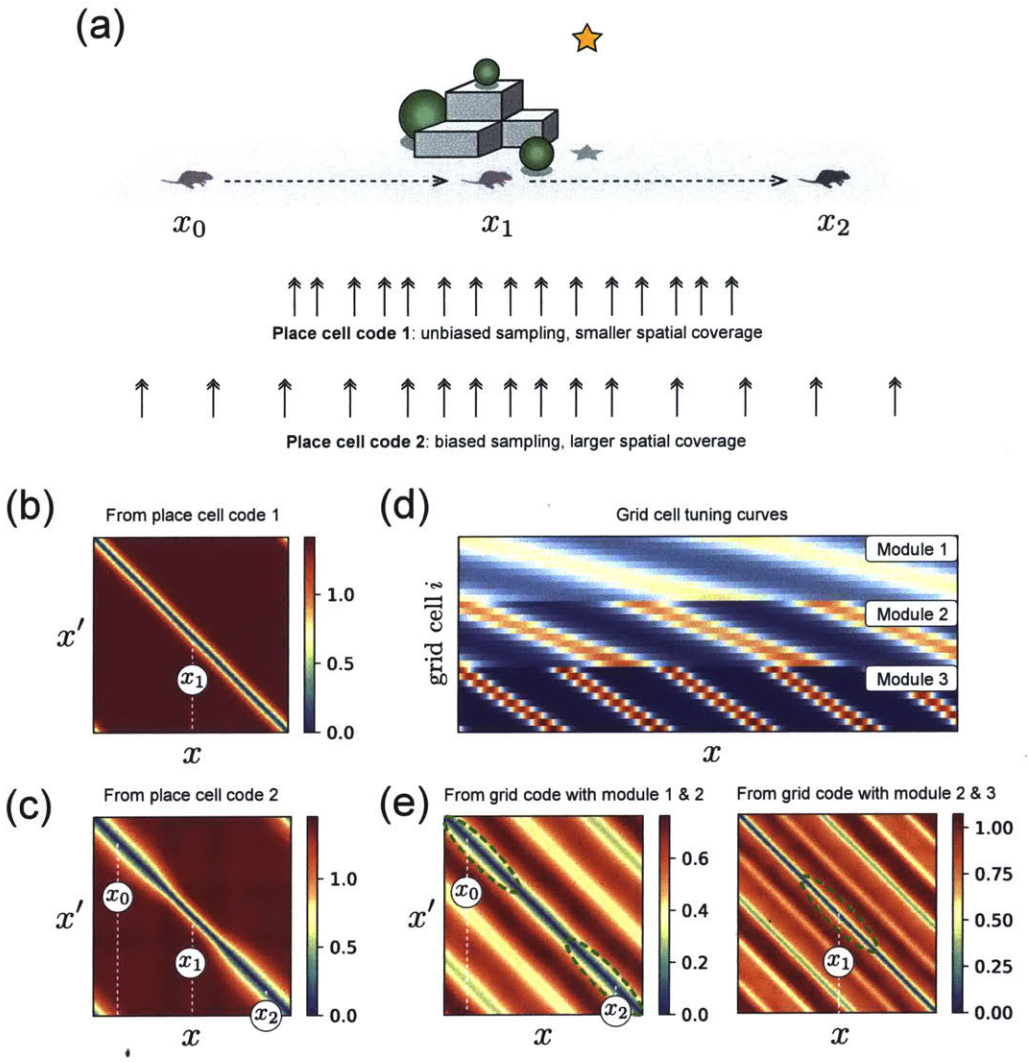

Figure 3.4: Place cells can flexibly allocate resources to create a topological code. **(a)** A biased landmark sampling rate can cover a larger spatial range with a finite amount of orthogonal codewords, while achieving demanded spatial resolution within the most behaviorally relevant region. **(b)** The distance matrix among the codewords from an unbiased place cell code is locally translationally invariant (TI) as can be seen in its constant diagonal structure. **(c)** The distance matrix from a biased place cell code breaks local TI condition, which illustrates the main feature of being a topological code is not to encode precisely the metric information. **(d)** A three-module grid cell code with different grid spacings can be used to guide the formation of a biased place cell code with locally different spatial resolutions. **(e)** The distance matrix of the assisting grid cell code composed with larger-spacing or smaller-spacing modules. The modular property of a grid cell code enables its compositional ability for which a new grid cell code composed out of only a few modules is still TI, and thereby still provide allocentric cues.

Vertices 1 and 3 in the resultant graph. In Figure 3.5(b), however, Codeword E overlaps with all four corner codewords such that it is better described as a vertex in the resultant graph. Figure 3.5(c) depicts how a location can even be encoded by

more than a single codeword. As a result, depending on the spatial trajectory going along either the path 1—3 or the path 2—4, different place cells can be activated. In which case, Codewords E and F contribute less to encode a position but more to encode a trajectory; consequently, this particular place code underlies a complex graph. This simple example therefore shows how a place cell code can fundamentally build an abstract graph out of a physical space—a topological code that helps an agent figure out which locations connect to which.

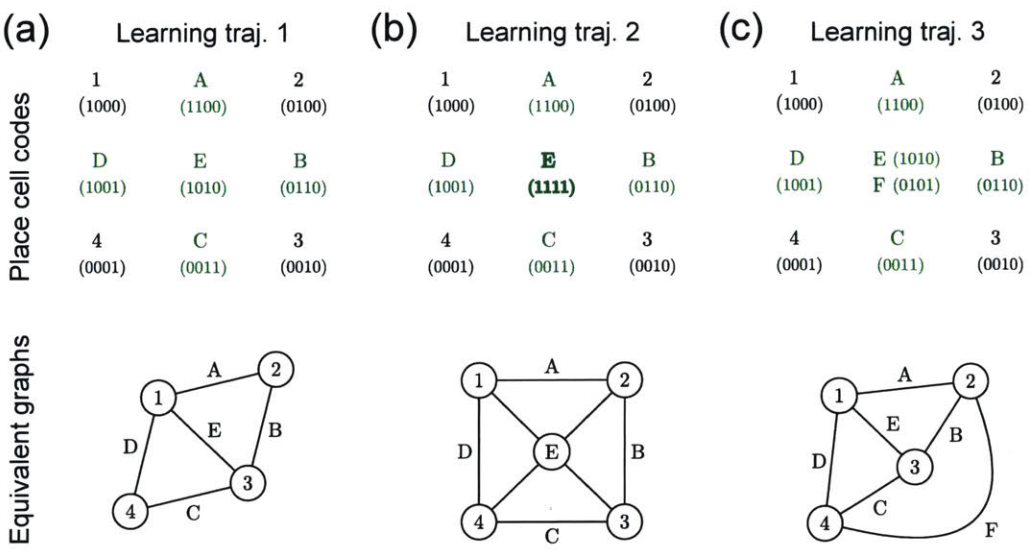

Figure 3.5: Different place cell codes can underlie various graphs in the same environment. **(a)** A code for nine locations that has four orthogonal codewords at corners (black) and five interpolating codewords (green) establishing the neighbor relations among them. Note that the Codeword E has only overlap with Codewords 1 and 3 but not Codewords 2 and 4, so that it underlies a graph shown below. **(b)** A different code for the same environment that has Codewords E connects Codeword 2 and 4 too, which results in a different graph. **(c)** A place cell code can also have more than one codeword per location which results in trajectory-dependent tuning curves and a complex underlying graph.

Topological codes or predictive maps as a higher-order learning

In earlier sections, we discuss some high-level functions of place cells that have been hypothesized. Most of them are based on the higher-order tuning properties beyond a uniform (unbiased) tiling of place fields in an environment. For example, place cells have been thought to form predictive maps (Stachenfeld et al., 2017) that stores necessary information for an agent to navigate towards rewards in a reinforcement learning setting (Gershman, 2018; Gardner et al., 2018). A predictive map—needing to encode reward information—is necessarily biased; In terms of our setting, locations near a reward can have either different landmark sampling rate or different sampling sparsity, or both. From experiments, most place fields stabilize rather quickly—a few minutes after a rat is placed in a novel environment. One can see this first stage of fast place field formation as a first-order learning—i.e., learning an unbiased map. After that, as the rat moves around, discovers more structures of the environment that could be relevant to the task, the first-order place fields are molded and gradually form a biased map. This slower learning of a biased map with more contextual information can thus be seen as a higher-order learning as an extension of the optimization

principle.

3.3 RNN sequential learnability on multi-environment landmark-prediction task

With the development of the optimal online learning algorithm, it is now straightforward to fulfill the core constraint in the optimization principle—i.e., to find a neural implementation that can perform the two algorithmic steps: 1) *optimal random landmark sampling* and 2) *online similarity matching*. Meanwhile, because place cells are required to operate independently once a code is learnt based on the *function complementarity*, it is best to use an RNN for such a neural implementation so that a place cell circuit can function as an independent path-integrator.

Before we discuss the explicit implementation of the two algorithmic steps (Section 3.5). It is important to investigate how the RNN dynamical property influences the ability of online learning. It's well known in the machine learning community that there is a major issue regarding using a single neural network to learn multiple tasks sequentially: the *catastrophic forgetting* (Goodfellow et al., 2013; Coop and Arel, 2013; Kirkpatrick et al., 2016). The cause of catastrophic forgetting is more of synaptic overrides than of exceeding capacity limit[4]. During the time in learning a new task, an RNN modifies its synaptic weight without concerning the performance of the earlier tasks.

Having said that, it is still possible to avoid catastrophic forgetting. As it turns out, the severity of synaptic overrides in an RNN largely depends on its dynamical property. The aim of this section is therefore to discover the type of RNN dynamics that can learn sequentially without catastrophic forgetting such that this insight can be transferred to the final neural implementation for completing this final piece in the optimization principle.

Despite not explicitly implementing the optimal online learning algorithm, we will see how an RNN—capable of sequential learning—naturally shares tuning properties with place cells. At the end of this section, I will introduce the second fundamental principle in this thesis: the *tuning-learnability correspondence*, which has an important implication about the nonspatial tuning properties of place cells that have been

[4]It is known that an RNN of 256 cells has the capacity to learn to navigate in a hundred different environments (Kanitscheider and Fiete, 2016)

abundantly observed in experiments (McNaughton et al., 1983; Mehta et al., 2000; Battaglia et al., 2004; Leutgeb et al., 2006).

3.3.1 Training scheme with random and balance RNN initialization

For simplicity, I use only a simple RNN layer for modeling place cells that receive a scalar sparse input for each landmark and a continuous velocity input. The grid cell inputs for the step of *random landmark sampling* are spared in this training scheme. Likewise, the step of *similarity matching* is replaced by a simpler training objective—an accurate prediction of an upcoming landmarks.

Setup of sequential learning scheme

I. Landmark prediction task. The task I adopted falls into the class of self-localization similar to those RNN training schemes in Chapter 2. During training, the agent random walks in a novel environment—a circular track with few landmarks—as shown in Figure 3.6(a,b) before a training in the next environments. Within an environment, the agent is demanded to predict the next landmark by gradually increasing the activity of an assigned landmark prediction neuron at the output end while approaching one. The supervised landmark prediction signals, $\mathbf{x}_k^{\text{pred}}$, are shown in Figure 3.6(d) in which one can see that the RNN has to raise the activity of a landmark prediction neuron—from roughly $60cm$ away—long before the encountering, which is only $5cm$ away when the RNN starts to receive landmark inputs: $\mathbf{x}_k$. During crossing the space between two landmarks—where there is no landmark inputs—the agent still needs to generate predictions continuously; it is therefore necessary for the RNN to path-integrate and know the agent's position at all time.

II. Dynamics, architecture, and training. The RNN has dynamics of a canonical neural network. The place cell layer has activity evolves according to

$$\mathbf{s}_P(t+1) = (1-\eta)\ \mathbf{s}_P(t) + \eta\ [W_{PP}\mathbf{s}_P(t) + b_P + U_v\mathbf{x}_v(t) + U_k\mathrm{x}_k(t)]_+, \tag{3.2}$$

where $\eta \equiv \frac{\Delta t}{\tau}$ is the discretization constant, $[u]_+ = \begin{cases} 0 & \text{if } u < 0 \\ 1 & \text{if } u \geqslant 0 \end{cases}$, x_v is velocity input from a correlated random walk, and x_k^t is a sparse landmark input. All hyperparame-

ters are listed in Appendix B.2. The activity of landmark predictions neurons follow a similar dynamics:

$$\mathbf{s}_L^{(k)}(t+1) = (1-\eta)\ \mathbf{s}_L^{(k)}(t) + \eta\ [W_{LP}\mathbf{s}_P(t) + b_L]_+. \tag{3.3}$$

In this scheme, since landmark predictions neurons only receive feedforward inputs from the place cell layer, they can as well be viewed as purely nonlinear readout layer. Though I preserved the natural decaying dynamics of a neuron here, one might not expect any qualitative difference in results if one simply uses a readout layer without dynamics, e.g. $\mathbf{s}_L^{(k)}(t) = [W_{LP}\mathbf{s}_P(t) + b_L]_+$.

The RNN is trained using standard gradient descent with backpropagation through time (BPTT) algorithm. The loss function in environment k at time t is defined as a L2-norm between landmark prediction signals $\mathbf{x}_k^{\text{pred}}(t)$ and the activity of landmark prediction neurons $\mathbf{s}_L^{(k)}(t)$:

$$L_k(t) \equiv \frac{1}{2N_L^{(k)}}\ \left|\mathbf{s}_L^{(k)}(t) - \mathbf{x}_k^{\text{pred}}(t)\right|^2, \tag{3.4}$$

where $N_L^{(k)}$ is the number of landmarks in Environment k. The total loss function for applying BPTT is

$$\text{Batch loss} = \sum_{t=t_0}^{t_0+T} L_k(t) \tag{3.5}$$

Each training session comes with a batch of 100 parallel trials, and the weight updates are the averages across all trials. A session last few-thousand to ten-thousand epochs depending on the error convergence. Each epoch last $10s$ (approximately the time for a rat to run 1 lep with maximum speed in a $6.4m$ long track). The truncated BPTT interval is $T = 1s$ which is approximately half the time for the rat to run from one landmark to the next. The significance of BPTT interval to the optimization principle will be discussed in Section 3.5.

III. Learning geometrically or topologically different environments. There are two types of environments in this sequential learning scheme. The RNN is trained in either a set of geometrically different or topologically different environments. Type I consists of ten environments as shown in Figure 3.6(a,b) with their difference defined by the track length, number of landmarks, and location of those landmarks; in other words, these environments are distinguishable from their geometrical differences. In these environments, all landmarks are distinctively specified by unique input weights

U_k. Type II consists also ten environments, as shown in Figure 3.9(a,b), but with few indistinguishable landmarks within a given environment. These environments are thereby distinguishable by both their geometry as well as their topology.

In either Type I or II environments, I trained the RNN in Environment 1 by learning W_{PP} and $W_{LP}^{(1)}$, and subsequently in Environment 2 after first training as illustrated in Figure 3.6(c). In the learning of Environment 2, I froze the trained environment-dependent readout weight $W_{LP}^{(1)}$, enable the update for $W_{LP}^{(2)}$, and keep on training W_{PP} in the meantime. The training will keep going in such a sequential fashion until completing all ten environments.

IV. Random and balanced initialization. In each training session of ten environments, I initialized the RNN using either the following two connectivity matrices as shown in Figure 3.7(a,d). For a random initialization, I used:

$$W_{PP} = \mathcal{G}(-2\beta, \beta), \tag{3.6}$$

where $\mathcal{G}$ draws a random gaussian matrix, and $\beta \ll 1$ is used for a stable initial learning with the RNN dynamics mostly reflects the instantaneous sensory inputs (Sompolinsky et al., 1988). For a balanced initialization, I used:

$$W_{PP} = \mathbf{I} - \beta, \tag{3.7}$$

where $\mathbf{I}$ is identity matrix. An RNN with a balanced initialization—similar to identity initialization reported in (Le et al., 2015)—has long time scale dynamics such that its activity integrates all sensory and self-motion inputs. We will be discussing the consequence of these initialization on the learning dynamics in subsequent sections.

V. Learning a new, and re-adapting to a familiar environment. During a training session of learning a new environment, the connectivity is modified using a constant learning rate lr_{train}; whereas during a test run in a familiar environment, I used a small learning rate: $lr_{\text{adapt}} = .1lr_{\text{train}}$ for a relatively small adjustment on the connectivity in order to require the original place fields.

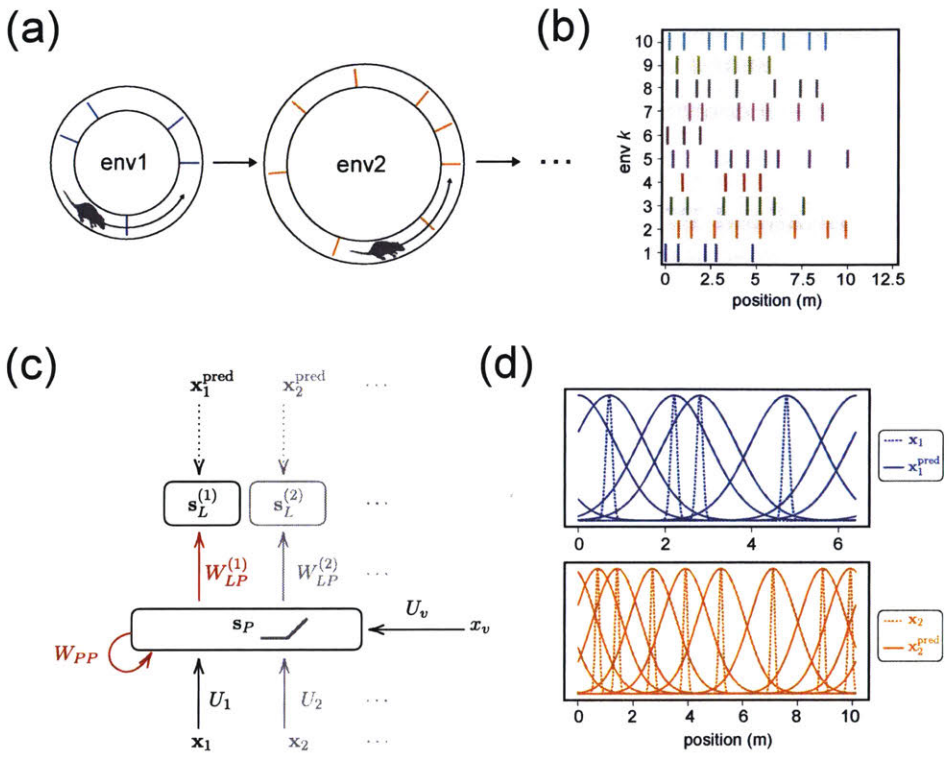

Figure 3.6: A sequential learning scheme of a landmark-prediction task that requires an RNN both to path-integrate and to avoid catastrophic forgetting. **(a)** An agent is trained to navigate in multiple environments one at a time. **(b)** Each environment is a circular track with a certain size and number of landmarks. All landmarks provide a unique inputs to place cell ensemble such that they are distinguishable. **(c)** The network for the task consists of a recurrent layer of place cells and a layer of landmark-prediction cells. Both the recurrent connections among place cells and the feedforward connections from place cells to landmark-prediction cells are trainable (marked red). The Place cells—that receive a stream of velocity inputs x_v and sparse landmark inputs x_i—need to learn an appropriate representation for the landmark-prediction cells to generate correct spatial signals that match the supervised signals. **(d)** Landmark inputs and landmark-prediction supervised signals in spatial coordinate for Environments 1 and 2. Note that the landmark-prediction signals are much broader than the landmark inputs so that the agent needs to path-integrate for generating correct outputs even when a landmark input is absent.

3.3.2 LD attractor dynamics exacerbates catastrophic forgetting

As described in the training scheme, because landmark prediction signals cover all those positions where landmark inputs are lacking, it is necessary for an RNN to integrate a velocity input in order to properly generate landmark predictions at all time. Consequently, the RNN has to learn a continuous attractor (Tsodyks, 2005; Samsonovich and McNaughton, 1997; Burak and Fiete, 2009; Kanitscheider and Fiete,

2017) that encodes the agent's position even though the position is not explicitly given as a supervised signal unlike another study that requires an LSTM network to generate explicit $x - y$ coordinate (Kanitscheider and Fiete, 2016).

A randomly initialized RNN learns an LD manifold

From the convergence based on the learning curve as shown in Figure 3.8(a), we see that a randomly initialized network can learn at least one environment. After training, the activity trajectory traced out a ring-like 1D manifold in the high-dimensional state space as shown in Figure 3.7(c). Because the low-dimensional (LD) nature of such a manifold, from this point onwards, I will call the corresponding RNN an *LD network*.

The LD network learnt such a circular manifold by adjusting its recurrent connectivity from the random initial weight: Figure 3.7(a). During learning, the activity trajectory started from a random point attractor—similar to a stored pattern in a Hopfield net[5] (Hopfield, 1984)—and gradually formed a one-dimensional continuous attractor as illustrated in Figure 3.7(b). The learning took a long time—see the blue curve of Figure 3.8(a)—because the LD network was, initially, only capable of a one-to-one mapping from the sensory inputs to the neural activity around this point attractor. Due to the lack of ability to integrate velocity input, the initial updates of the recurrent connectivity at different time tended to cancel each other out. Only when a set of points beyond the original point attractor became relatively stable, the learning sped up—as can be seen in the blue curve of Figure 3.8(a) around 2000 epochs.

A subsequent learning trajectory highly depends on the last learnt LD manifold

After learning Environment 1, the LD network succeeded in learning the subsequent ones as shown in the learning curves in Figure 3.8(a) in which a successful learning is defined as the error asymptotes around or below $10cm$—the width of a landmark—indicated in the green shaded region in Figure 3.8(a). One caveat, though, is that the three learning curves looks very different—which indicates a sign of very different learning dynamics within individual environments beyond just an effect caused by the environmental differences. To understand this, one can compare the manifold

[5]Or more appropriately, a continuous Hopfield network model—also known as Grossberg additive model (Grossberg, 1988)

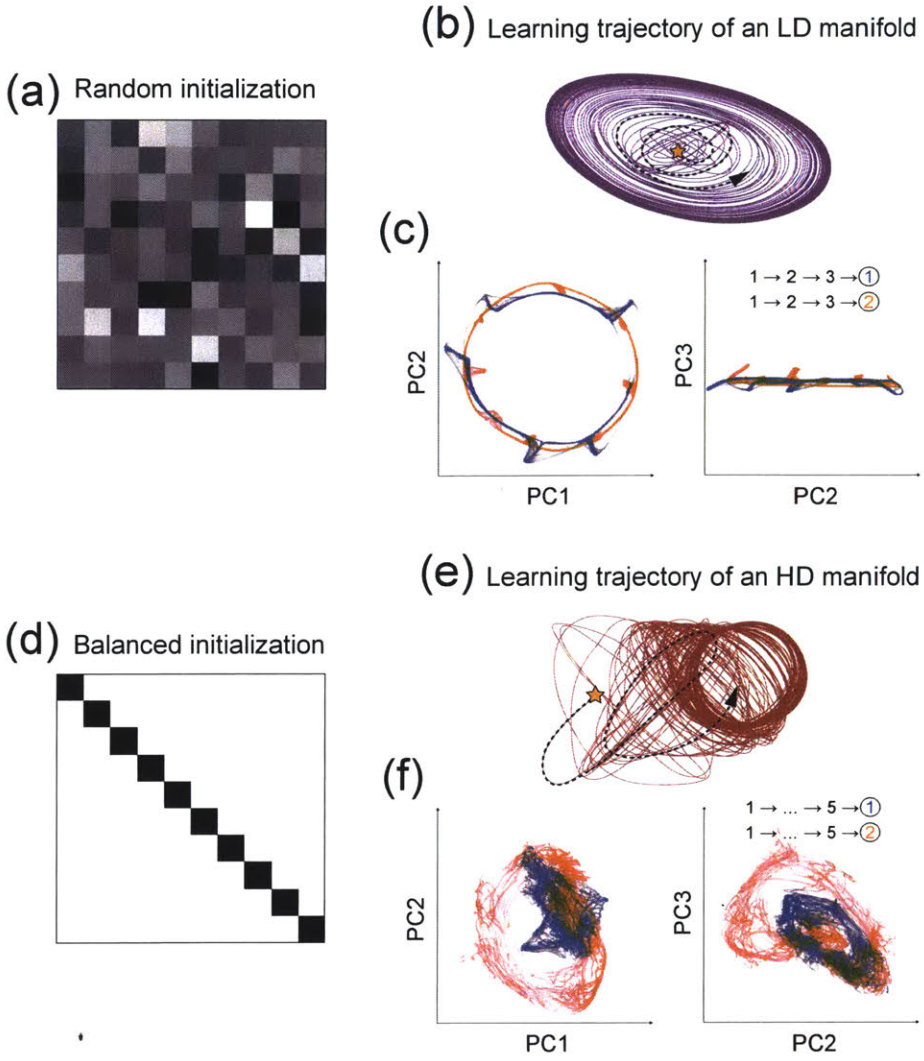

Figure 3.7: Qualitatively different learning trajectories of LD and HD networks are results of their distinct dynamics set by two classes of initial recurrent connectivity. **(a)** A randomly initialized recurrent connectivity matrix that will result in a low-dimensional (LD) manifold after learning. **(b)** The illustrated learning trajectory of an LD manifold. The learning starts from a random point attractor, and the LD network gradually develops a ring attractor for path-integration. **(c)** State trajectories of Environment 1 and 2 projected to the first three principal components after consecutively training in three environments. The fact that the second manifold are on top of the first one indicates the reuse of first learnt manifold when learning the second environment. **(d)** A balanced-initialized recurrent connectivity that will result in a high-dimensional (HD) manifold after learning. **(e)** The illustrated learning trajectory of an HD manifold. The learning starts from an inconsistent state trajectory on a flat energy landscape, and a consistent trajectory is gradually developed for those states that were more frequently visited. **(f)** State trajectories after training in five environments. The manifolds are HD with a little overlap which implies the latest learning is largely independent of the existing learnt manifolds.

of Environment 1 and 2, plotted together in Figure 3.7(c) after learning Environment 1 to 3. The fact that the two manifolds plotted with the shared principal components are on top of each other suggests that there is only one ring-like manifold developed after training in these three environments (See Appendix B.4 for a comparison of manifolds in all three environments).

It is worth pointing out that the *track length*, *number of landmarks*, and *position of landmarks* are all different for these three environments; if one trains three randomly initialized networks on these environments separately, the chance for their manifolds to overlap should be almost zero. This simple thought experiment explains why the learning curves of Environment 2 and 3 are so different from that of Environment 1 in Figure 3.8(a). Because the LD network possessed only a single strong ring-like attractor after learning just one environment, the learning trajectory of the next environment had to start from those states on the attractor given that they are the only stable states in this current RNN. The subsequent learning dynamics was therefore greatly influenced by the existing LD manifold, and consequently the manifold was modified to suit the task in Environment 2. Similarly, the manifold was modified again after learning Environment 3.

A strong attractor dynamics accelerates the catastrophic forgetting

The fact that an LD network is incapable of learning multiple manifolds is problematic. This drastic manifold modification in an LD network underlies an important issue in the machine learning community: the *catastrophic forgetting* (Goodfellow et al., 2013; Coop and Arel, 2013). To show that a modified manifold is no longer compatible for using in an earlier learnt environment, I tested the performance of an LD network—after learning only three environments—to navigate in the earlier two environments. In a test run, a small learning rate—*one-tenth* of the value in a training session—is applied for the network to "re-adapt" to a familiar environment. The results are shown in Figure 3.8(b). One can see that the LD network could not revert to the original performance in Environment 1 even after an extensive period of time much longer than the original training time for achieving desirable performance[6].

The catastrophic forgetting problem was extensively studied in Hopfield networks

[6]Note that the LD network was able to recall Environment 2 after learning Environment 3 potentially because that these two environments are more similar in their track length and number of landmarks; thereby the manifolds learnt in Environment 2 can be adopted in Environment 3 with minor changes and vice versa (Note that also the learning curves of Environment 3 is fast converging after learning Environment 2)

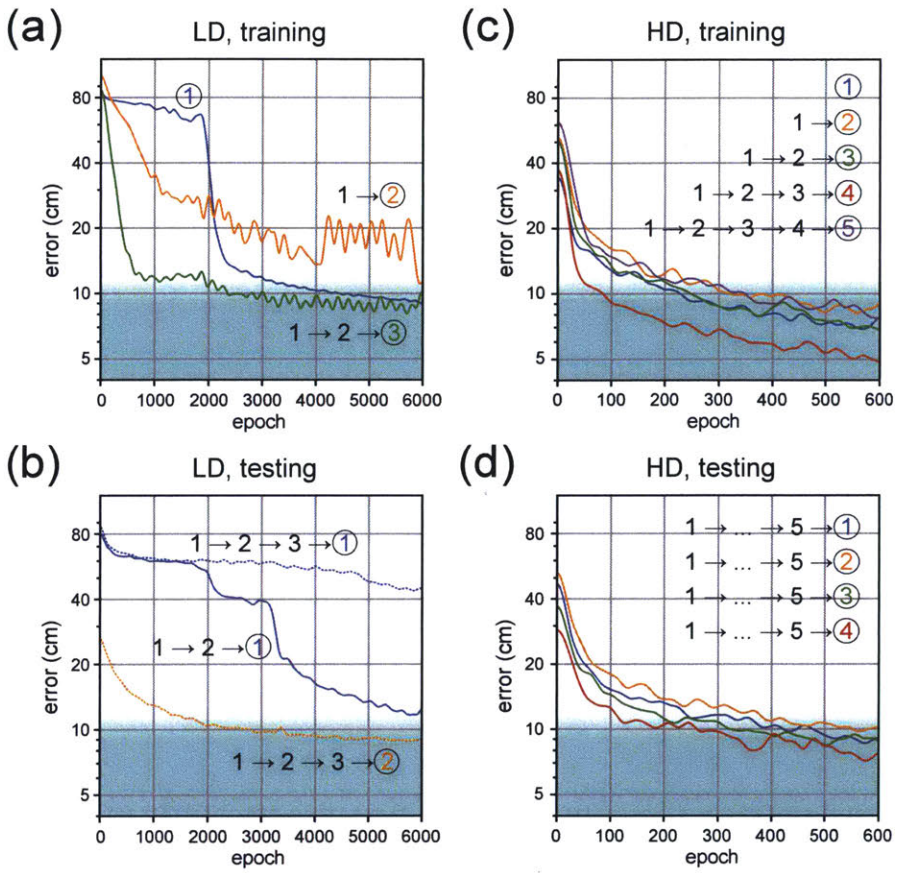

Figure 3.8: Qualitatively different learning and testing curves show the presence of catastrophic forgetting in an LD network and the absence of it in an HD network. **(a)** Learning curves of an LD network in first three environments. A successful learning is defined when the desirable prediction error (indicated by a green shaded region) is reached. The fact that a rapid convergence only happened after 2000 epochs in Environment 1 (blue) implies that it takes a long time to develop a ring attractor from an initial point attractor. The vastly different learning curves of Environment 2 (orange) implies the influence from the first learnt ring attractor. **(b)** Testing curves of the LD network with *one-tenth* of the original learning rate. The failures of going back to the desirable performance within the original learning time reveal an exacerbated catastrophic forgetting in an LD network. **(c,d)** Learning and testing curves for an HD networks. The learning time for an HD network is more than ten-fold less than that in an LD network. The qualitatively identical—monotonically decreasing—learning curves for all five environments imply that there is little influence of the existing manifolds on the learning of a new one. The fast convergence in the testing phase also shows that an HD network is much less influenced by catastrophic forgetting.

in which the quality of memories degrade drastically if the number of stored patterns exceeds the network capacity (Hertz et al., 1991). However, before the runaway memory loss by crossing the capacity limit, the forgetting problem in a Hopfield network is, in fact, manageable, and the memory only degrades gradually due to the

inevitable small overlaps among stored patterns. This simple example of the forgetting mechanism really makes it more obvious the severity of catastrophic forgetting in an LD network in which it has the capacity of learning merely a single environment. This extreme version of catastrophic forgetting highlights just how important an RNN dynamics is when it comes to sequential learning. Because an ongoing RNN dynamics directly determines the course of learning (i.e., modifying RNN connectivity and molding manifolds), an RNN with a strong attractor dynamics suffers from an accelerated modification of the earlier stored knowledge; hence an exacerbated catastrophic forgetting.

3.3.3 HD dynamics enables sequential learnability

Now we understand how, in an LD network, a strong attractor dynamics underlies the forgetting mechanism. These results thus motivate another class of RNN operating in entirely different dynamical regime.

Balanced initialization: learning from a flat energy landscape

If an RNN starts its learning from a balanced initialization:

$$W_{PP} = \mathbf{I} - \beta, \tag{3.8}$$

where $\mathbf{I}$ is identity matrix and $0 \leqslant \beta \leqslant 1$, then the initial RNN dynamics goes as follow:

$$\begin{aligned} \tau \frac{d\mathbf{s}_P}{dt} &= -\mathbf{s}_P + [W_{PP}\mathbf{s}_P + b_P + U\mathbf{x}]_+ \\ &= -\mathbf{s}_P + [\mathbf{s}_P - \beta N_P \langle \mathbf{s}_P \rangle + b_P + U\mathbf{x}]_+ \\ &\approx -\beta N_P \langle \mathbf{s}_P \rangle + b_P + U\mathbf{x}, \end{aligned} \tag{3.9}$$

where $\langle \mathbf{s}_P \rangle \equiv \frac{1}{N_P} \sum_i s_{P,i}(t)$ is the average activity, and the argument inside the rectified linear unit is assumed to be positive for the most time. If we further assume that $\langle \mathbf{s}_P \rangle \approx 1$ is constant in time and set

$$b_P = \beta N_P \langle \mathbf{s}_P \rangle, \tag{3.10}$$

the initial RNN becomes an integrator:

$$\tau \mathbf{s}_P = \int U\mathbf{x}\, dt \tag{3.11}$$

If there is no external inputs, any states $\mathbf{s}_P$ is initially stable given that $\dot{\mathbf{s}}_P = 0$. The learning dynamics of a balanced-initialized RNN thereby contrasts that of a randomly-initialized one; that is, it carves channels out of a flat energy landscape, establishes local instability for constructing a stable manifold (On the other hand, an LD network—from a randomly initialization—constructing a stable manifold by reshaping and expanding a local point of energy minimum.)

It is worth pointing out that there exists a continuous set of RNN solutions for a flat energy landscape, i.e. $\{W_{PP} \equiv \mathbf{I} - \beta \mid 0 \leqslant \beta \leqslant 1\}$. The eigenvalues of W_{PP} equal 1 for all except one of them equal to $1 - \beta N_P$—with the corresponding eigenvector $\mathbf{s}_P \propto (1, \ldots, 1)^T$. The fact that all other $N_P - 1$ eigenvectors have eigenvalues that are independent of β implies the similar integrating dynamics as explained above. Two polar extremes of $\beta = 0$ or 1 can be biologically interpreted as an RNN with strong synaptic self-excitation that cancel out cell decays, or an RNN without self-excitation but with strong uniform background inhibition, respectively. In the training scheme, I initialized an RNN using a small $\beta = 2/N_P$. But one should expect a drift of β if there is no special constraint that prefer one extreme than the other.

Learning an HD manifold

The learning trajectory of an HD manifold is illustrated in Figure 3.7(e). Since all states are initially stable, one can expect a rather chaotic trajectory in the state space in the early stage of learning. As the learning progresses, those states revisited more frequently become lower in energy than their surrounding states, and a much more stable manifold was thereby gradually developed. Figure 3.7(f) shows the final learnt manifolds of Environment 1 and 2 after sequentially learning five environments. One can clearly see that these manifolds are qualitatively different from an LD manifold in two major aspects: 1) the underlying dimension of the manifold is much larger than 1—the spatial dimension of a task, and consequently 2) a given position is not encoded by a single state on the manifold but by a set of states spanning in all dimensions. Because the high-dimensionality of such a manifold, from this point onwards, I will call the corresponding RNN an HD network. It is important to note that despite the fuzziness of an HD manifold seems to produce a source of uncertainty

in representation for decoding position, the performance on the task is nonetheless high[7]

An HD network learns and recalls five environments in sequence

Another important takeaway from these two manifolds plotted in Figure 3.7(f) is that they have unambiguously different shapes, which implies that the subsequent learning is largely not constrained to the existing manifolds. The ability to learn a different manifold in a novel environment greatly reduces the effect of manifold overrides happened in an LD network, and hence greatly mitigates the catastrophic forgetting. This drastic improvement on sequential learnability can be, again, attributed to the distinctive learning dynamics of an HD network in which a large fraction of states remained stable—or rather untouched—even after learning a few environments. A new manifold can thus be developed out of those unused states without much overlapping with the existing ones (Note that the fuzziness of these manifolds also explains the reduction the overlaps, in which a quantitative estimate (Equation (3.14)) of the overlap between two manifolds is done from computing the tuning overlaps)

The training and testing results of an HD network that learnt five environments is shown in Figure 3.8(c,d) respectively. The fact that—all five learning curves converged in a qualitatively similar way and the time it takes for a successful training was around the same order of magnitude for all environments—indicates the lack of a strong attractor to influence the course of learning in the early stage. Note that the learning time in the case of an HD network is more than ten times faster (learning time is on the scale of 100 epochs, or 1000 s = 17 mins) than that in the case of an LD network (learning time is on the scale of 2000 epochs, or 5 hrs and 33 mins). Moreover, in a test run, the performance quickly revert to the desirable after roughly the same time it took during the earlier training sessions regardless of using only one-tenth of the original learning rate. These results show that an HD network is capable of learning multiple environments in sequence without the catastrophic forgetting like an LD network.

[7] A similar demonstration on how an uncertain representation can give rise to a stable performance can be found in this study (Druckmann and Chklovskii, 2012)

An HD network can learn topologically different environments

Motivated by the fact that—in a trained HD network—there is no simple one-to-one mapping from *position* to *state*, it is interesting to ask how an HD network will learn its representation if there are ambiguous landmarks in an environment. For example, as shown in Figure 3.9(b), Environment 2 contains three identical landmarks such that the original circular track might well be an eight-shape or a clover-leaf-shape track with identical landmarks appear only once at the center from the observer's point of view. Meanwhile, because there is no constraint about assigning the same state to these identical landmarks, the topology—i.e., the underlying graph of the representation—of a learnt HD manifold needs not respect the topology of an environment.

In this training, I again compared the performance of an HD network with an LD network. From Figure 3.9(c), we can see that an LD network had a much harder time learning a topologically non-trivial environment. For example, after the first successful learning on Environment 1, the network was totally incapable of learning Environment 2, yet was able to learn Environment 3. Oddly enough, an untrained LD network was, nevertheless, not able to learn Environment 3 without learning Environment 1 first. Also, if an LD network first learns Environment 2 (which is topologically more complex), it can then learn Environment 1 or 3 (which is topologically simpler). The corresponding learnt manifolds were plotted in Figure B.3(a,b). Note that the manifold—after successfully learning a topologically non-trivial environment, i.e., Environment 2—was highly distorted and developed a subloop. All in all, these results reconfirm the strong dependence of a learning trajectory on an existing manifold with strong attractor dynamics in an LD network.

As for an HD network, the learning and testing trajectory did not differ from those of non-topological environment. They all converged in the qualitatively same way. A closer investigation on the learnt manifolds—in Figure B.3(c)—showed that they still possess the two main features that define an HD network: 1) these manifolds are high-dimensional and 2) they do not encode a position using just a single state. In principle, as long as the trajectory on an HD manifold can consistently produce landmark prediction signals, it doesn't matter if these locations with identical landmarks are encoded as the same state.

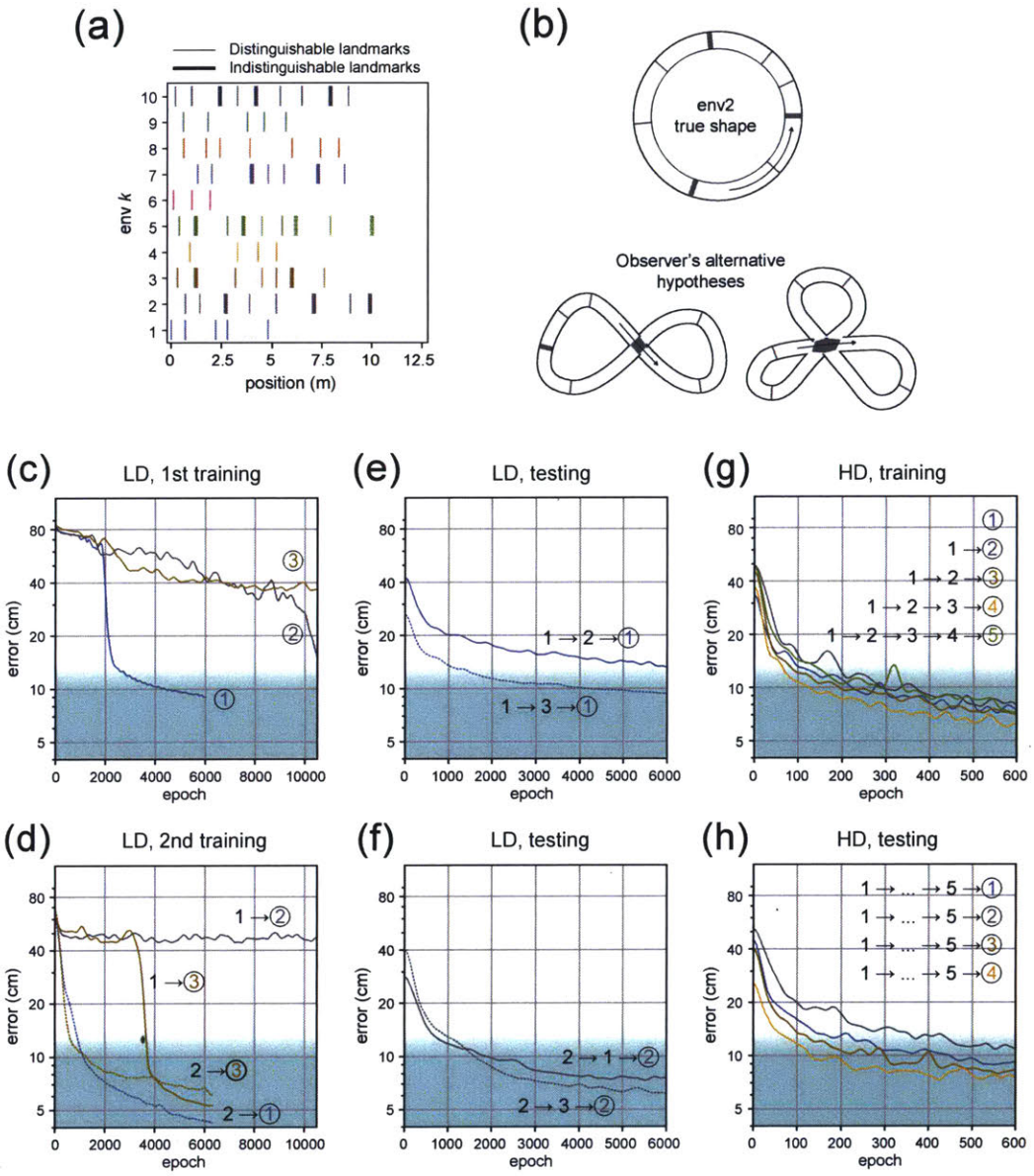

Figure 3.9: An HD network can learn topological environments with the same efficiency while an LD network struggles from topological incompatibility. **(a)** Environments with indistinguishable landmarks are topologically nontrivial, or simply, *topological*. **(b)** Environment 2 is topological (with three ambiguous landmarks) in the sense that a different environment shape than a circle can also produce identical landmark inputs that are indistinguishable from the observer's viewpoint. *(c-f)* Learning and testing curves for an LD network with an emphasis on that a failure of learning happens when the topology of the consecutive environment is incompatible to that of the currently existing manifold. *(g-h)* Learning and testing curves for an HD network. The lack of difference between the curves implies that there is no qualitative difference between the learning in a topologically trivial or non-trivial environment.

An HD network is capable of learning ten environments

At last, I also pushed an HD network to learn ten environments—both only geometrically different or topologically different ones—with the results shown in Figure B.4.

Although it did take longer for the network to come back to a decent performance in comparison to learning only five environments, the tuning memory degraded in a non-catastrophic way. This process of a slow memory loss is very much an analogue of storing random patterns in a Hopfield network as discussed earlier, in which the memory degradation is manageable as long as the number of stored items is within the network capacity limit. The degraded tuning memory is plotted in Figure B.4(d,h) where the overlap—between 1) the tuning curves when first trained in Environment k and 2) the recalled tuning curves in a test run after learning all ten environments—are calculated. One can see that the degradation is roughly linear as a more distant tuning curve have less overlap with the recalled one.

3.3.4 HD dynamics is inherently nonautonomous without intrinsic representation

To better understand the inner workings of both LD and HD network, I ran relaxation simulations with 2000 random initial states on a network that has been trained on multiple environments; the results are shown in Figure 3.10. During the relaxation, both velocity and landmark inputs were turned off, and the state iterated according to the following autonomous dynamical equation:

$$\tau \frac{d\mathbf{s}_P}{dt} = -\mathbf{s}_P + [W_{PP}\mathbf{s}_P + b_P]_+, \tag{3.12}$$

where $[u]_+ = \begin{cases} 0 & \text{if } u < 0 \\ 1 & \text{if } u \geqslant 0 \end{cases}$. A relaxation simulation lasted for half a second and the last states were projected onto the first two principal components computed from all 2000 last states.

An LD network learns an intrinsic representation as a ring-like continuous attractor

In an LD network one can see that these random initial states quickly settled down to a ring attractor within fraction of a second. The instantaneous relaxation time—indicated in color—are around $.1s$. The shorter the relaxation time, the faster the decay—in the direction orthogonal to a manifold, and hence the more stable the manifold. Note that the relaxation time plotted is computed for the second longest decaying mode

(See Appendix B.6 for details), because the longest mode is parallel to the manifold and thus not informative regarding the stability of a continuous attractor. For an LD network there was always only a single manifold despite learning multiple environments; this also confirmed the modification on the original, Environment 1, manifold as discussed earlier. The inner workings of the LD network thus agrees with previous studies on path-integration with a continuous attractor (Tsodyks, 2005; Samsonovich and McNaughton, 1997; Burak and Fiete, 2009; Kanitscheider and Fiete, 2017). At algorithmic level, an LD network builds a continuous set of stable states as a spatial representation that can exist without external inputs, or *autonomously*. The nonautonomous part of an LD dynamics then serves as a perturbation for performing path-integration—i.e., the velocity input updates a current state to its neighboring state within the set, and the landmark inputs pin-pointing particular states within the set serves as error correction against diffusion[8] (Burak and Fiete, 2012)].

An HD network needs not learn a stable representation to path-integrate

We have known that—from Figure 3.7(f)—an HD manifold enables path-integration in spite of not respecting the underlying dimensionality of the task relevant variable (in the landmark-prediction task, it is a 1D spatial variable). Given the high-dimensionality of this manifold at any given position, no one stable state should encode a single position. However, one should still expect some weak stability for states near the manifold. Indeed, this is what I saw when running the relaxation simulation on a trained HD network as shown in Figure 3.10(b). The relaxation time is more than ten times longer than that of a trained LD network. Surprisingly, even after the network has been trained on five environments, these states were still, to a large extent, not attracted to a general region—that supposedly resembling a loop—close to the HD manifold when external inputs were present. These results also imply that all states beyond the fuzzy manifold region are *metastable* with a long relaxation time, and thereby are still available for learning a new non-overlapping manifold for a subsequent task.

[8]Instead of spikes as a noise source like (Burak and Fiete, 2012), the diffusion can still be caused by—a finite discretization time, non-vanishing learning rate, occasional abrupt turn in random walk, not-perfectly smooth continuous attractor, etc.—even when an explicit noise source is absent.

Coding stability only exists when a consistent series of external inputs are present

Algorithmically, the results from an HD network suggest a much less intuitive—at the algorithmic level—solution for path-integration. Unlike an LD network, an HD network is no longer using the nonautonomous part of its dynamics as perturbation to update the state within a pre-existing stable representation. Instead, the line that separates autonomous and nonautonomous parts of the dynamics vanishes, and the HD dynamics as a whole is inherently nonautonomous. This essential feature of the HD dynamics implies that the coding stability necessary for performing a task can only exist when external inputs are present; more importantly, the landmark inputs that defines an environment need to be consistently present at the correct locations in a random walk trajectory. For example, if one landmark is removed or replaced after training, the ongoing state will react to the vacancy that could send the trajectory off the original manifold. In an extreme case, if the network is still learning, it could well recognize this one-landmark-removed environment as a novel environment, and thereby exhibit a global remapping. More regarding this dynamical-constraint-induced remapping will be discussed in Section 3.4.3. Overall, the inner workings of an HD network do not follow a simple algorithmic solution like an LD network, which contrast the conventional representation-first computation (Hertz et al., 1991).

3.3.5 Place fields, remapping, and the tuning-learnability correspondence

From the above discussion, we know how different the autonomous dynamics of an HD network is compared to a continuous attractor dynamics in an LD network: The lack of strong attractor dynamics largely contributes to the sequential learnability of an HD network. And the abundance of metastable states—even after learning a few environments—allows an HD network to learn a new manifold largely unconstrained by the existing ones. Next, we will study these networks' nonautonomous dynamics—when there are external inputs—by investigating their *position* and *velocity* tuning within an environment, as well as comparing these tuning curves across environments. Surprisingly, an HD network is not only capable of learning like place cells, i.e., having sequential learnability without catastrophic forgetting, but also share similar tuning properties with them.

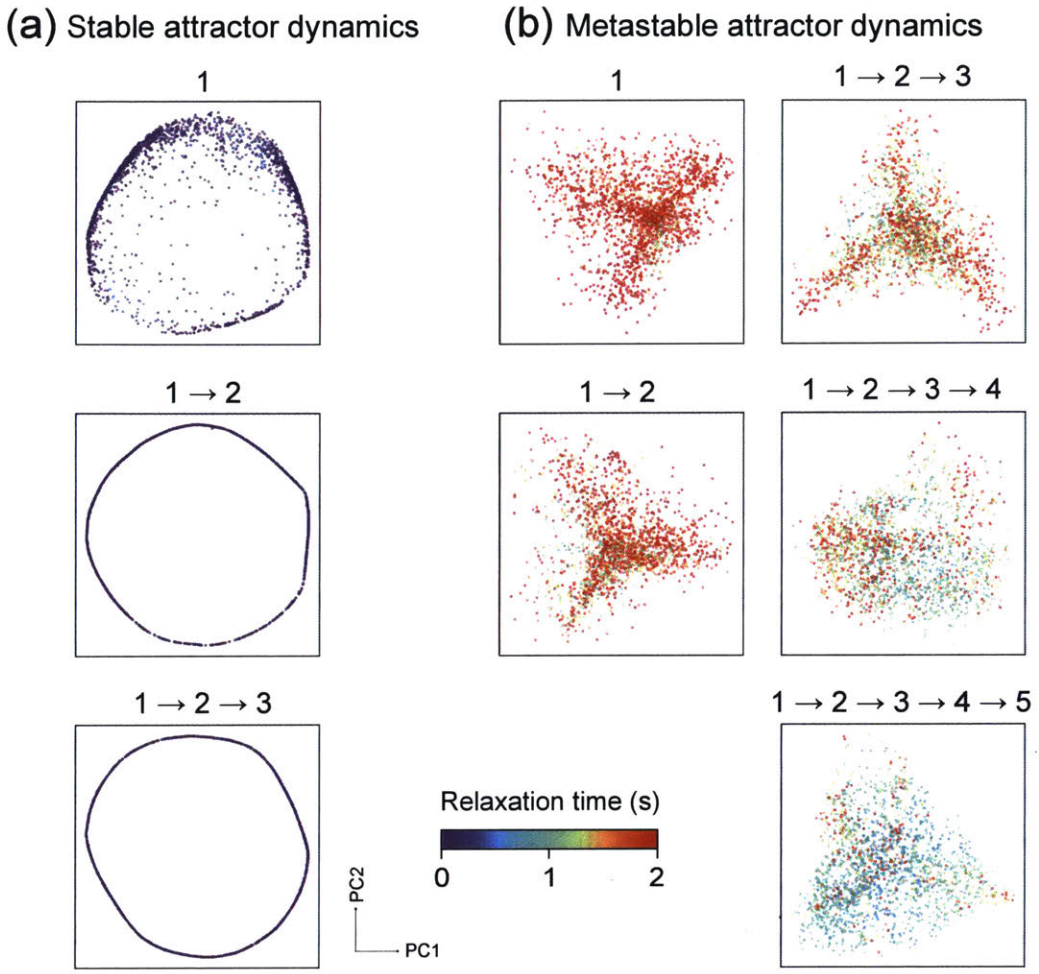

Figure 3.10: Metastable attractor dynamics of an HD network implies a different mechanism for path-integration from the conventional representation-first approach. A relaxation simulation for a trained network were run from 2000 random initial states separately without external inputs. The simulation lasted for $.5s$; all last states are projected to the first and second principal components and plotted together. *(a)* The fact that, for an LD network, all random initial states quickly settle down to form a continuous ring of stable states within $.5s$ shows the existence of one (and only one) stable ring attractor after training in one, two, or three environments. *(b)* For an HD network, there is no stable low-dimensional representation even after learning five environment. These results suggest that an HD network path-integrates in a very different way from the conventional representation-first implementation approach, and the consistent representation only show up when the external inputs are present. An HD network thereby has inherently a *nonautonomous* dynamics.

An HD network has narrow and directional tuning curves

The example tuning curves from Environment 2 plotted in Figure 3.11(b,e) are computed as a heat map with 10cm bin size in the spatial axis and 16 equal sized bins

along velocity axis:

$$z_i^{(k)}(r,v) \equiv \sum_{t \in \text{ test epochs}} \{s_{P,i}(t) \mid t = t_{r,v}\}, \tag{3.13}$$

where $t_{r,v}$ is moments when position and velocity equals r and v respectively. The 24 out of 256 selected tuning curves are from the cells receiving either positive, near-zero, or negative feedforward-weighted velocity input, respectively. The comparison reveals two major differences between an LD and HD networks: 1) an HD network has much narrower and localized tuning curves, and 2) an HD network possesses prominent directional tuning curves. Curiously, these two tuning properties coincide with place cells (McNaughton et al., 1983) especially in a one-dimensional track (Mehta et al., 2000; Battaglia et al., 2004; Kjelstrup et al., 2008).

An HD network has narrow and directional tuning curves

Next, I computed the tuning correlation between Environment k and l—a measure to quantify similarity between two maps:

$$D_{kl} \equiv \sum_{i,j} C_{ij}^{(k)} \; C_{ij}^{(l)}, \tag{3.14}$$

where $\left\{C_{ij}^{(k)}\right\}$ (properly normalized) is the pairwise correlation matrix for Environment k that summarize the spatial tuning correlation among all cells:

$$C_{ij}^{(k)} \equiv \int y_i^{(k)}(r) \; y_j^{(k)}(r) \; dr \tag{3.15}$$

with the marginal tuning curves: $y_i^{(k)}(r) \equiv \int z_i^{(k)}(r,v) \; dv$.

For an LD network, I computed D_{kl} after learning three environments. The result in fig. 3.11(c) shows a near a-hundred-percent preservation on pairwise tuning correlation, a tuning property similar to grid cells (Trettel, 2017; Yoon et al., 2013; Fyhn et al., 2007)—which only realign though a constant shift and rotation within a gird modules. This result reconfirms a complete reusage of the first learnt manifold for a further learning in an LD network.

For an HD network, D_{kl} computed for five environments in fig. 3.11(f)—or ten environments in Figure B.4(c)—are much smaller indicating a complete change in pairwise tuning relations moving from one environment to another—hence a global

remapping similar to that observed in place cells (Wills et al., 2005; Colgin et al., 2008; Alme et al., 2014)

An HD manifold is not a 2D manifold that explicitly encodes velocity

In the earlier discussion, I quickly touched upon the advantage of high-dimensional manifolds in sequential learnability—i.e., these manifolds circumvent the catastrophic forgetting issue due to a small overlap among them as shown in Figure 3.7(f). Here I investigated what variables are these manifolds encode by coloring them according to either the agent's instantaneous velocity or position—as shown in Figure 3.11(d). It reveals how an HD network actually encodes these variables using such an exotic continuous manifold[9] for performing the task. In Figure 3.11(d), the position-colored manifold shows that the position is encoded in the longitudinal direction (along the fuzzy loop). Notably this loop is neither a 1D manifold like that of an LD networks nor a flat 2D manifold with the shape of a ribbon. Instead, the loop is truly high-dimensional given that the fuzzy structure is visible in multiple projections using different principal components. The clear separation of the colors on the manifold suggests the decodability; together with proper feedforward projection weights to the landmark prediction cells, a trained HD network is capable of predicting upcoming landmarks very accurately.

When the manifold is velocity-colored, one also sees a clear color separation in the transverse directions (orthogonal to the loop). However, it is important to keep in mind that this observation only implies the existence of a pronounced velocity tuning. The HD network was not trained to accurately encode velocity, thereby it's not at all clear if such a high-dimensional arrangement of states—resembling a fuzzy manifold—is useful for decoding velocity. Conceptually, if the task would require an accurate velocity decoding too, it is most easy to dedicate a unique transverse direction for encoding velocity—given a position—such that the manifold is locally two-dimensional and its global structure resembles a ribbon embedded in high-dimensional state space.

[9]This is in contrast to the LD network which learnt a one-to-one mapping from position to state on a 1D ring-like manifold as shown in Figure 3.11(a); the mixing in the color blue and red highlights that these cells are lack of velocity tuning.

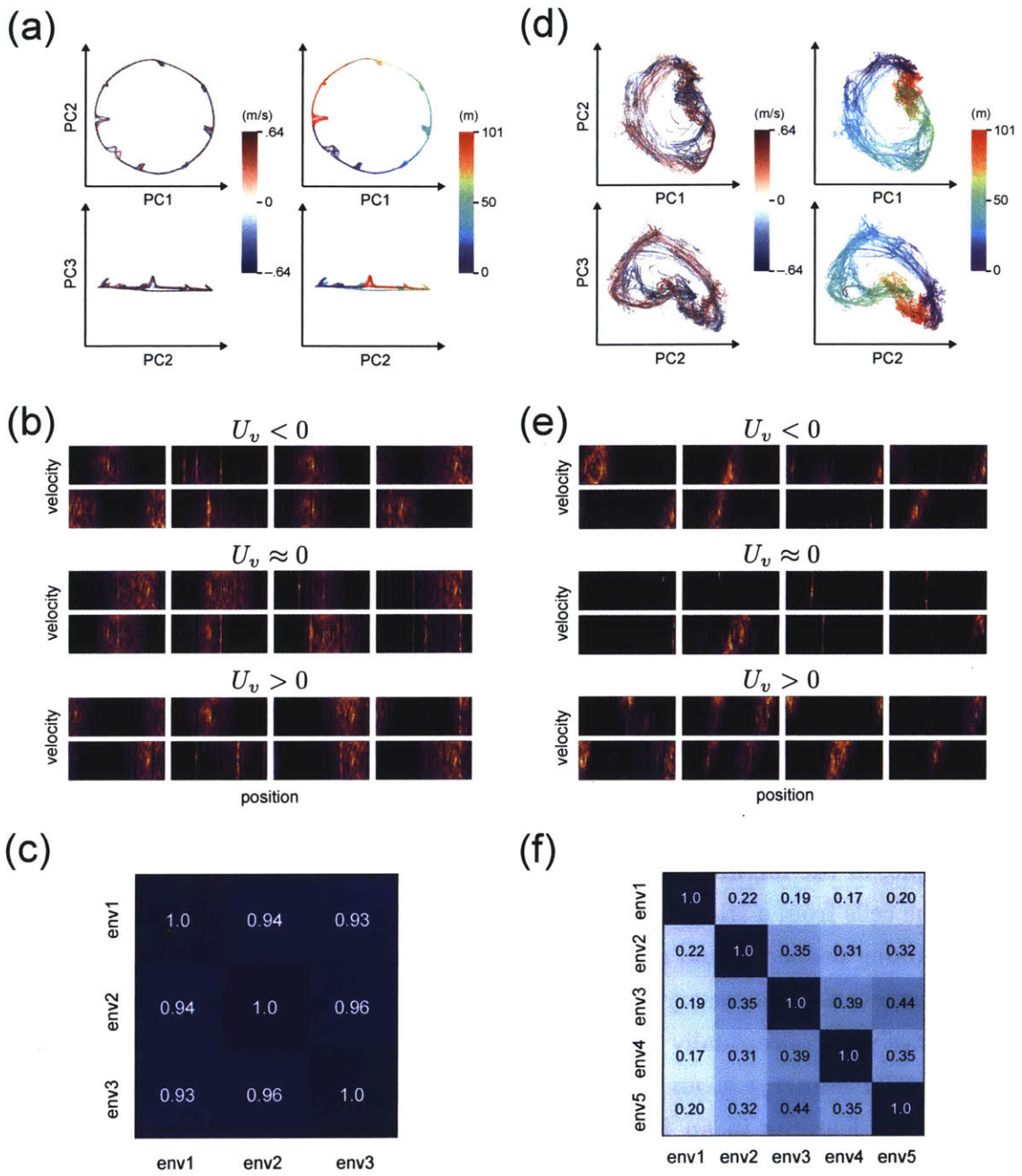

Figure 3.11: Metastable and inherently nonautonomous dynamics of an HD networks reveals how place cells might learn and path-integrate. (Caption continued on the following page.)

The tuning-learnability correspondence

The above comparison of the tuning properties between a learnt LD and HD networks uncovers a fundamental principle between tuning curves and sequential learnability, which I named the *tuning-learnability correspondence*:

Figure 3.11: Metastable and inherently nonautonomous dynamics of an HD networks reveals how place cells might learn and path-integrate. (a-c) are results for an LD network trained in three environments, while (d-f) are for an HD network trained in 5 environments. **(a)** The manifold of an LD network when navigating Environment 2. The manifold being colored by either velocity or position shows that it would only take a pure position code for solving such a landmark-prediction task. **(b)** Joint velocity-position tuning curves of LD network from three populations of cells that receive position, near-zero, and negative feedforward projections for velocity inputs. The cells are only position tuned and tend to have wider tuning curves. **(c)** The cross-environment tuning correlation matrix (D_{kl} between any pairs of Environment k and l, defined in the main text) shows that the cofiring structure of an LD network is preserved across different environments, which is a tuning property similar to that of grid cells. **(d)** The colored manifold of a trained HD network shows a seemingly redundant velocity coding unnecessary for solving the task. **(e)** The cells in an HD network show a clear conjunctive tuning property for encoding both position and velocity. **(f)** The cross-environment correlation matrix shows that the cofiring structure is not preserved in an HD network: a result known as global remapping in place cells. The fact that an HD network shares both similar tuning properties and sequential learnability with place cells suggests that a place cell ensemble might learn and path-integrate through a nonautonomous metastable attractor dynamics.

> *For a simple RNN to possess a sequential learnability without catastrophic forgetting, the cells are necessarily conjunctively tuned beyond task relevant variables.*

For example, in the landmark-prediction task, the task relevant variable is position, and yet the cells from a trained HD network are also highly velocity tuned. This conjunctive tuning property need not exist at algorithmic level (Marr and Poggio, 1976), but it is nonetheless necessary at implementation level, consequently resulting in a less intuitive and conceptually more complex algorithmic solution[10]. In contrast to a non-sequential learner, e.g. an LD network that builds a strong attractor for position encoding, the learning in an HD network is to achieve a minimal stability that is sufficient to perform a task[11]. In so doing, a learnt manifold—with no constraint to respect the underlying dimensionality of the task relevant variables—tends to occupy as many dimensions as possible in the state space. The learning trajectory of an HD manifold in state space is largely at random initially—as illustrated in Figure 3.7(e); as learning progresses, some trajectories repeatedly revisited become part of the final

[10] A simpler algorithmic solution for a sequential task would be to encode only the task-relevant variables but in a subpopulation of cells. In such a case, the algorithm can just allocates another population of unused cells.

[11] A balanced initialized network has zero stability—for which all states are stable at first—so that a tiny perturbation could send a desirable state off track. Under the course of learning, the stability of manifold grows and then stops growing after reaching desirable performance.

manifold. This learnt manifold can thereby be seen as a bundle of various 1D trajectories—as shown in Figure 3.11(d)—consequently giving rise to conjunctive tuning curves of both task relevant and irrelevant variables.

This study demonstrated that it is crucial to take dynamics seriously from the beginning in order to understand the process of learning in the brain. In terms of Marr's levels of analysis, one should study the algorithmic and implementation levels together (See the discussion in Section 1.3.4), or in an interleaving way, such that the dynamical constraint becomes a major part of the study to guide a discovery of novel algorithmic solution.

3.4 CA3 recurrent network as a learnable path-integrator

Earlier studies on CA3 cells in the hippocampus led to few theories on functional roles based on this region's rich recurrent collaterals (recurrent CA3–CA3 synapses) (Witter and Amaral, 2004). The most studied functions include *pattern completion*, *pattern separation*, and *path-integration*. While the pattern completion or separation may be studied separately in a nonspatial task (MacDonald et al., 2011; Rolls, 2013), a successful path-integration requires that all three functions work together. However, unlike pattern completion or separation, learning recurrent connectivity in performing path-integration suffers greatly from the *catastrophic forgetting* (as discussed in the last section). This issue is largely unresolved because the earlier theoretical approaches—named *representation-first paradigm*[12] (Hertz et al., 1991)—have been focusing on the modeling via a hand-crafted RNN that embeds a desirable representation as point or continuous attractors, and a strong attractor dynamics is the root cause for an accelerated catastrophic forgetting.

In this section, I will briefly introduce these three hypothesized functions of CA3 place cells with an emphasis on the issue regarding learning recurrent connectivity that arises from the need for a stable attractor dynamics. I will discuss an alternative solution for performing path-integration without using the conventional representation-first model called the multichart attractor network. This novel solution for path-integration is the high-dimensional (HD) network, discussed in Section 3.3, which does not have strong attractor dynamics and can learn multiple environments

[12]The common theme for such an approach is to, first, build a minimal or idealist representation, and, second, to manipulate the state transition within representation for complete a task or achieving a function; hence the *representation-first paradigm*.

sequentially. Finally, I will discuss the experimental implications and predictions made by this novel solution.

3.4.1 Pattern completion, pattern separation, and path-integration in CA3

In representation-first paradigm, one creates attractor states in an RNN to represent either certain memory items (an item can be visual, audial, olfactory, etc.), or some hidden variables that are relevant to behaviors (e.g. representing position for navigation). The reason for using attractor states can be twofold: 1) It is for stability such that the noise (e.g. stochasticity from poisson spikes) in a circuit does not wash off the desirable state at a given time. 2) It is for memory recalls from partial cues (recalling having seen an apple on the desk just by seeing the empty desk). Stability, or noise resistance, is required for all three functions of CA3 place cells we will discuss below, whereas a memory recall from partial cues is most relevant to pattern completion.

Pattern completion

Pattern completion has been thought as an important function of CA3 because the hippocampus can store and recall an episodic memory rapidly from only partial or noisy cues—an ability known as autoassociative memory (Rolls, 2013; Guzman et al., 2016)—at any time within an episode. The requirement of fast memory retrieval (Treves et al., 1997) can be fulfilled with an RNN that stores stable point attractors. It is thus hypothesized that CA3, with recurrent connections, is the main subregion that performs pattern completion.

In parallel to episodic memory, pattern completion in CA3 can also be considered as autoassociative memories for all locations within a familiar environment. It is similar to storing random patterns in a Hopfield network but for a set of continuous locations. Surprisingly, even when a rat traverses to the region where the sensory inputs are seriously deprived, stable place fields are still retained (Leutgeb and Leutgeb, 2007). This ability that requires more than a simple memory recall from sensory cues is called path-integration which will be discussed later.

Pattern separation

In order to store and recall many different memory items in CA3, representations for the memory as point attractors should be separable under noise and ideally are orthogonal to each other. To have many orthogonal representations, they have to be sparse—i.e., only few active cells to represent a memory item. If so, the chance for two memory items to have overlapping representations is greatly reduced even when these two memories only differ by a little in perception. This property to have dissimilar representations for two perceptually similar memories is called pattern separation (Leutgeb and Leutgeb, 2007; Rolls, 2013). One way to achieve such a transformation from similar stimuli to sparse representations is to use *competitive learning* (Rumelhart and Zipser, 1985) such that only a small subset of cells, being a winning group, will eventually be excited by a particular stimulus.

There are many variations to implement competitive learning; in the context of neural networks, one can use mutual inhibitions among all cells while learning the feed-forward weights from stimuli (input nodes) to individual cells. The overall strength of mutual inhibitions—implemented as inhibitory recurrent connections—sets the sparsity of a representation; the cells thereby are required to compete in a winner-take-all (WTA) dynamics for representing a stimulus (Rolls et al., 2006). Having the recurrent connectivity, CA3 subregion could perform pattern separation on its own. But from the anatomical evidence (Marr, 1971), CA3 seems to achieve pattern separation together with dentate gyrus (DG): an upstream subregion in hippocampus.

In the DG-CA3 duo system, DG provides sparse representations that are then projected to CA3 in a nearly one-to-one fashion though mossy fibers[13]. The activity pattern in CA3, in response to the DG input, is then engraved as a point attractor through Hebbian learning on the corresponding CA3–CA3 synapses. With the help of DG for providing sparse representations of stimuli, CA3 recurrent connectivity is free for the additional purpose, and the hippocampus in fact accomplishes both pattern separation and completion at the same time.

Path-integration

The most common functional name for a hippocampal cell is place cell. Not surprisingly, they got this name for their ability to represent the animal's location. In a cue

[13]One CA3 cell out of 300,000 receives roughly only 46 mossy fiber inputs, whereas it receives inputs from 12,000 other CA3 cells (Rolls, 2013)

rich environment, theoretically, place cells can transform sensory inputs at any location into a set of sparse representations. This reliable one-to-one mapping between a (partial or noisy) sensory cue to a sparse spatial representation can be done by combining both pattern separation and pattern completion as discussed above. However, amazingly, place cells can reliably represent locations even when the sensory inputs are deprived (O'Keefe and Nadel, 1978). In such a case, the animal needs to rely on its self-motion cues to perform path-integration.

As discussed in Section 3.3, to perform path-integration, a neural network needs to represent a continuous variable as its state (activity pattern) and to update the state using self-motion cues. The conventional approach is to first build a continuous attractor with each state on it representing a unique position, and then properly incorporate sensory and self-motion cues by perturbatively modifying feedforward connections (Samsonovich and McNaughton, 1997). To build a continuous attractor, one needs recurrent connectivity. It is thus only sensible that CA3 place cells has been hypothesized to play such a functional role. However, with so many functional roles an ensemble of CA3 cells could play, it is unclear if CA3 place cells truly perform path-integration by themselves. An alternative hypothesis could be that CA3 place cells only receive spatial inputs from another path-integrator—e.g. from the entorhinal grid cells (Solstad et al., 2006)—such that it appears to be path-integrating.

Without falling into any particular debate regarding how plausible the CA3 cell ensemble can path-integrate alone, there is nonetheless a fundamental issue for learning such a path-integrator in an RNN—i.e., the catastrophic forgetting. Because this issue directly relates to the main functional role of place cells: having a learnable representation, it has to be resolved before claiming any possibility of CA3 place cells being a standalone path-integrator. Below, I will discuss how this issue is partially resolved in the multichart attractor network model (McNaughton et al., 1996; Samsonovich and McNaughton, 1997), what the remaining problems are, and how it can be resolved based on the results from the learning in an high-dimensional (HD) network discussed in Section 3.3.

3.4.2 Metastable attractor dynamics enables a learnable path-integrator

Earlier studies on modeling path-integration in CA3 cells focused on answering these questions: 1) how to store multiple continuous attractors in an RNN? (McNaughton

et al., 1996; Samsonovich and McNaughton, 1997; Tsodyks, 2005), 2) how many such continuous attractor can be stored? (Battaglia and Treves, 1998), and 3) how to reconcile both continuous attractors and point attractors (Rolls et al., 2006) such that a sensory cue can evoke a spatial memory (Leutgeb et al., 2006; McHugh et al., 2007), or vice versa? Each of these questions has been largely answered by the multichart attractor network model (Samsonovich and McNaughton, 1997). As for the big question regarding sequential learning of representations in multiple environments, a partial answer is also given in which a multichart attractor network learns feedforward projections that bind sensory inputs to pre-existing continuous attractors in the network. However, from both biological or theoretical viewpoint, there are still many important unresolved issues regarding the existence of pre-stored continuous attractors and the plausibility of only learning feedforward projections. For that, I will address these issues below and discuss how the HD network model introduced in Section 3.3 can resolve them.

Remaining Issues on the multichart attractor network being a model for CA3 place cells

Biological perspectives A multichart attractor network requires a prewiring of its recurrent connectivity that cannot be learnt through experience. The main reason is the same for that an LD network cannot learn a new continuous attractor without substantially modifying the earlier learnt ones. As a model for place cells, however, it is demanded to learn a novel representation continuously. To circumvent this major problem, McNaugton et al. proposed to keep recurrent connectivity fixed and only to learn the feedforward projections (McNaughton et al., 1996; Samsonovich and McNaughton, 1997). In so doing, a new representation is learnt not by creating a new continuous attractor, but by binding sensory inputs of a given environment to one of the pre-existing attractors. Despite the fact that the experiment already suggests that CA3 recurrent connections is plastic (Nakazawa et al., 2002), I will discuss below three main reasons why multichart attractor network is not biological plausible.

The first issue concerns the plausibility of wiring up such a pre-existing connectivity. A multichart attractor network has connectivity that appears to be random, yet is topographical in an individual chart. A simple construction of a K-chart connectivity matrix is given as

$$W_{K \text{ charts}} \propto \sum_{k=1}^{K} W_k, \tag{3.16}$$

where

$$W_k \propto \exp\left(-(r_{ij}^k)^2/2\sigma^2\right)$$

with r_{ij}^k being the distance between i- and j-th cells in the arrangement of k-th chart. r_{ij}^k is designed to be k-th random permutation of $r_{ij}^0 \equiv |i-j|^2$ such that the total connectivity looks random when $K \gg 1$ yet precisely wired in such a way that the partial connectivity W_k is topographical when cells are arranged according to k-th chart's indices. It is thereby hard to imagine a biological blueprint capable of wiring up $W_{K \text{ charts}}$ even via some unknown developmental processes as mentioned by McNaughton et al. (1996). However, it is worth to stress that it is still plausible for such a process to wire up a single chart with the full connectivity to be topographic. For this reason, the conjecture made in Chapter 2—regarding grid cell circuit being developed rather than learnt—is still valid because a grid cell ensemble forms only a single chart such that one can choose $r_{ij}^{\text{grid}} \equiv |i-j|^2$ to be topographically arranged. The evidence for such a topographical connectivity has been observed in grid cells (Gu et al., 2018).

The second issue concerns the existence of a necessary controlled switch among pre-existing continuous attractors, or charts. While it might be easy to imagine a central control that allocates a recently unused chart when the rat is artificially moved to a new environment (which explains global remapping) in a laboratory, it is not at all clear how such a control works in a more realistic (large) foraging environment[14]. For example, one may ask, what spontaneously triggers a controlled switch when a chart is used up? How does this central control monitor an unknown environment such that it knows how to best allocate unused resources? What happen if a chart is only half-used; will the other half be reallocated by the central control in another environment? And if so, how does the central control know not to override the half that has been used? All these problems are crucial and needed to be addressed theoretically before claiming plausibility of the multichart attractor network model.

Lastly, without external inputs, all charts should be excited equally likely. This implies that the cofiring structure of place cells during sleep should not confine to only the recently used charts. However, most experiments have suggested otherwise, for which the cofiring structure most resembles only the recent experience but not prior to that (Wilson and McNaughton, 1994) and decays rapidly during subsequent sleep (Kudrimoti et al., 1999). These experiment results imply that, not just the

[14]There is no evidence of spontaneous chart switching in a very large environment (Rich et al., 2014), the place fields equivalently form a large continuous chart.

feedforward sensory projections, the underlying attractors and thereby the recurrent connections are modified throughout experiences.

Theoretical perspectives Theoretically, the first question is what determines the geometry of the pre-stored charts. Should they be either 1D or 2D continuous attractors, or both? This is a rather important question given that the capacity for a multichart attractor network is low (Battaglia and Treves, 1998)—e.g. $N_{\text{2D charts}} \sim 10^{-3} N_{\text{cells}}$ within a plausible parameter regime—such that available states on each chart are expensive. It is therefore very much inefficient to map a 1D environment (e.g. a circular track) to a 2D continuous attractor. Even having a multichart attractor network that stores a mixture of 1D and 2D continuous attractors, a sophisticated controlled switch is still needed for determining which type of attractor is best allocated for binding an ongoing environment, which is not plausible as we just discussed above. On the other hand, the HD network model, to be discussed next, does not pre-store attractors, but instead it builds necessary geometry (and topology) as the agent explores a novel environment.

Following the above question with a tentative assumption that all charts are identical (so that there is no hassle of choosing which), there are still issues regarding how to handle different size of environments. A chart potentially can adapt to a range of different environment sizes if the gain of self-motion cues—e.g. as a velocity input—can be learnt. Unfortunately, in order for a multichart attractor network to path-integrate precisely and stably, a single common velocity input has to be well calibrated to be used by all charts and cannot be easily altered without compromising precision (Samsonovich and McNaughton, 1997). The connections regarding self-motion cues thereby have to be pre-wired along with the pre-existing continuous attractors[15]. This constraint regarding precise calibration for self-motion cues could potentially be relaxed if there are many sets of customizable feedforward connections for individual charts; yet again, a controlled switch is required for jumping from one set to another.

Taken together, either the biological or theoretical implausibility is from the constraint of needing a fixed recurrent connectivity in a multichart attractor network model. To resolve the issues above, a vastly different network model with learnable recurrent connectivity is required.

[15]The reasons that this pre-wiring is not required for sensory cues are that 1) sensory cues are both allocentric and sparse in time such that there is no accumulation of error in the learning process, and 2) each sensory cue is high-dimensional and can be environment-specific such that there is no need to calibrate it for all the charts.

Multiple representations can be learnt rapidly in metastable attractor dynamics

In a recent study, Kanitscheider and Fiete (2016) trained a network with 256 LSTM units to simultaneously self-localize in 100 2D environments with nearly optimal accuracy. This result is a clear indication that an ultra-high-capacity solution exists (on the order of one cell per environment) that must operate in a very different dynamical regime from a multichart attractor network. It has thus shown a possibility to release the need for building stable continuous attractors to perform path-integration.

From my results with the HD network introduced in Section 3.3, I have shown that a sequentially trained HD network does not have any stable intrinsic representation when external inputs are absent. In spite of that, it can accurately path-integrate with the presence of sensory and self-motion cues. It is this metastable attractor dynamics of an HD network that enables a learnable representation via modifying recurrent connections (See Sections 3.3.3 to 3.3.5 for details). In such a dynamical regime, a stable manifold can be learnt without ever creating a stable continuous attractor, and such a manifold can be flexibly adopted for any geometry or topology of an environment (See Section 3.3.3 for a discussion on learning topological environments).

One critique from (Samsonovich and McNaughton, 1997) regarding learning recurrent connections in CA3 on the fly is the potentially long learning time. From experiments, a new set of place fields in CA3 can be learnt and stabilized very rapidly on the order of 10 mins in a small environment (Wilson and McNaughton, 1993). While it is true that it takes a long time to learn a continuous attractor—e.g., It takes 5 hours and 33 mins for an LD network to learn the first ring attractor as shown in Figure 3.8(a), it only takes 17 mins for an HD networks to learn the first environment as shown in Figure 3.8(c) because it needs not build an attractor for path-integration. To sum up, the metastable attractor dynamics enables a rapidly and sequentially learnable RNN that captures all the essential static or dynamic tuning properties of CA3 place cells.

3.4.3 Experimental implications and predictions

With the issues regarding learning recurrent connectivity being resolved via the HD network model, It is now fully plausible that CA3 place cells play a functional role as a standalone path-integrator. Some important experimental implications and predic-

tions about CA3 place cells that come along with this novel network model are listed below.

Without upstream grid cell inputs, place fields remain stable in a familiar environment even when sensory inputs are deprived. The prediction comes with the hypothesis that CA3 place cells can operate as an isolated path-integrator. It also follows the theoretical stance of this thesis: the function complementarity which assumes that grid and place cells have complementary coding properties such that a place cell code cannot simply be a transformation of a grid cell code.

Cofiring structure during sleep resembles recent experience to a certain extent. As discussed above regarding how a multichart attractor network cannot account for such an effect, an HD network—despite having no internal stable attractors—still develop a general region of stronger attraction in state space, which we may tentatively call: a metastable attractor as shown in Figure 3.10(b). This broad attraction region of a metastable attractor is where all learnt manifolds lie. In that sense, those states on the fuzzy manifolds during awake will have higher chance to be reactivated during sleep. A supporting result in Figure B.4(d,h) in Figure B.4 shows how a tuning memory gradually degrade as the learning progresses due to an accumulation of small synaptic overrides. This result indicates that the last learnt manifold will overlap the most with the territory of the metastable attractor and consequently the states on it will have the most chance to be reactivated during sleep. In sum, this tuning curve drifting effect—also observed in experiments (Ziv et al., 2013)—accompanying with the recent-experience-biased reactivation during sleep can all be well explained by the learning dynamics of the HD network model.

Conjunctive tuning curves always happen even when the underlying task requires no such variables. From the *tuning-learnability correspondence* discussed in Section 3.3.5, we know that the formation of place fields in a novel environment is highly history dependent; that is, in the process of learning, all the nudges on the state trajectory from other task irrelevant sensory inputs (e.g. speed, colors, textures of the floor, sounds, smells) will shape the final learnt manifold to some extent (See Figure 3.11(d) as an example of velocity being a task irrelevant variable). As a result, the cells in an HD network will be tuned to these extra variables regardless of their necessity to the task.

A global remapping can happen even when a single sensory cue in a familiar environment is changed. This property has been observed in experiments from very early on (Muller et al., 1987). And while there are many speculations about the

exact function of remapping in various circumstances, I suggest that the dynamical constraint from CA3 recurrent connectivity causes global remapping when the input inconsistency in a familiar environment exceeds some threshold. To be more specific, we now know that an HD network processes a consistent manifold only when both sensory and self-motion cues are present due to its metastable attractor dynamics. If, say, one out of five landmark sensory inputs are removed after the HD network is trained, the state trajectory will start to go off at the location where the sensory input used to be. If this deviation is not corrected to stay within the basin of the next sensory cues, the state trajectory will not come back to the previously learnt manifold and develop a new one that are largely orthogonal; hence a global remapping.

A global remapping can be caused by non-perceptual cues associated with a sudden change in behavior. It is possible for the hippocampus to receive a non-perceptual signal during a sudden behavioral change. For example, an internal signal that switches from a random foraging to a goal-directed searching (Dupret et al., 2010). In such a case, the HD network model predicts that global remapping will happen because the agent necessarily learns two manifolds due to the difference in the inputs of the two behaviors. On the contrary, a model with sensory inputs being the largest driver for stable place fields would predict such a global remapping to be highly unlikely. For future studies, quantifying and developing a theory for such *dynamics-induced global remapping* could be an important topic that accounts for the role of recurrent dynamical constraints—not just functional constraints—on remapping, which is missing in the current literature.

3.5 RNN implementation of optimal online learning algorithm

Having the knowledge that an RNN can sequentially learn like place cells in a simple landmark prediction task, we are finally ready to fulfill the core constraints in the optimization principle. The key is to find an RNN learning scheme that can approximate—as much as it can—an optimal online learning algorithm introduced in Section 3.2: the *online sparse manifold transformation.* Specifically the learning scheme needs to perform 1) optimal random landmark sampling and 2) online similarity matching. In this section, I will discuss a candidate scheme with some variations. At last, I will demonstrate how such an RNN realization of this optimal learning al-

gorithm naturally predicts a biased field propensity across environments (Rich et al., 2014; Lee et al., 2019)—a result that seems to contradict the optimization principle for place cells at first glance.

3.5.1 k-winner-take-all RNN as a baseline sequential learner

From Section 3.2.2, we know that an optimal random landmark sampling is both unbiased and sparse. To achieve this, one can adopt a grid-assist architecture that uses an innately unbiased grid cell code as a sampler such that every cell can be drawn with equal probability. Specifically, the recurrent and feedforward connections have to be translationally invariant so that all cells behave the same way.

A k-winner-take-all RNN can unbiasedly sample a sparse random codeword

For an RNN to achieve an unbiased and sparse sampling, a k-winner-take-all (kWTA) network with translationally invariant connectivity is a proper candidate:

$$W_{kWTA} \equiv (\alpha + \beta)\, \mathbf{I} - \beta, \tag{3.17}$$

such that the place cell activity evolves according to

$$\tau \frac{d\mathbf{s}_P(t)}{dt} = -\mathbf{s}_P(t) + g[W_{kWTA}\mathbf{s}_P(t) + b_P] \tag{3.18}$$

Despite the simplicity, there are several degrees of freedom to be finefuned for a desirable behavior of this kWTA network, that includes two hyperparameters: α and β, and a nonlinear function: $g[u]$. The name, kWTA network, I adopt here is meant to stress that a sampled codewords has more than a single winner; it needs not have exactly k winners towards convergence. A candidate nonlinear function can be either sigmoidal (Grossberg, 1973; Majani et al., 1989) or rectified-linear (Xie et al., 2002; Kriener et al., 2017). As an example, if one uses a rectified linear unit: $g[u] = [u]_+ = \begin{cases} 0 & \text{if } u < 0 \\ 1 & \text{if } u \geqslant 0 \end{cases}$, the condition $\alpha < 1$ ensures a bounded solution, and $\alpha+\beta \lesssim 1$ ensures a convergence of more than one winner (Xie et al., 2002). In practice, a sampling time interval could be too short to use a asymptotic codeword $\mathbf{s}_P$ $(t \to \infty)$ as a random landmark; for which, it is more appropriate to flexibly finetune α and β

such that a transient codeword fits into a desirable criteria.

A kWTA RNN can operate near HD dynamical regime

In addition to the finetuning for a desirable sampling sparsity, it is most important to keep the kWTA RNN operating in the HD dynamical regime in order to maintain its sequential learnability as discussed in Section 3.3. For doing so, $\alpha + \beta \approx 1$ such that this RNN stay close to the balanced initialization:

$$W_{kWTA} \approx \mathbf{I} - \beta \tag{3.19}$$

Uniform grid-to-place projection: an unbiased dimensionality expansion

Lastly, we need to choose the feedforward weight U_{PG} for grid cell inputs $\{\mathbf{s}_G(t_l) \mid l = 1, 2, \cdots\}$, where t_l is a series of discrete sampling times. Because of the high-capacity nature of a grid cell code (See Chapter 2), any two adjacent sampled codewords are nearly orthogonal: $\mathbf{s}_G(t_l) \cdot \mathbf{s}_G(t_{l+1}) \approx 0$, which is a desirable property for optimal random landmark sampling; namely, for sampling orthogonal codewords. It is thereby optimal to use $U_{PG} = \mathbf{I}$ for preserving all information from grid cell inputs if the number of place cells $\mathbf{s}_P$ matches that of grid cells $\mathbf{s}_G$. In practice, however, there is a dimensionality expansion from N_G grid cells to N_P place cells such that

$$m \equiv N_P / N_G > 1 \tag{3.20}$$

It is therefore optimal to have one grid cell projects to m place cells in a non-overlapping and translationally invariant way. For simplicity, one may use:

$$U_{PG} \equiv \{u_{i=\ell m:(\ell+1)m,\ j=\ell} = 1 \mid \ell = 0, 1, \cdots, N_G - 1\}, \tag{3.21}$$

which converges to identity matrix when $m = 1$ or $N_P = N_G$. Alternatively, one may use a smooth version via a gaussian function with slight overlaps between adjacent cell groups:

$$U_{PG} \equiv \left\{u_{i=\ell m:(\ell+2)m,\ j=\ell} = e^{-(i-(\ell+1)m)^2/2m^2} \mid \ell = 0, 1, \cdots, N_G - 1\right\} \tag{3.22}$$

which approximately converges to identity matrix when $m = 1$. Both feedforward projects in Equations (3.21) and (3.22) are translationally invariant.

Sparse transformation via the MEC-DG-CA3 pathway

The dimensionality expansion proposed above is based on the need for an unbiased sampling of maximal number of orthogonal codewords. The input to place cells is therefore not sparse in general; that is, $\mathbf{h}_G \equiv U_{PG}\mathbf{s}_G$ has the same sparsity as a grid cell codeword. To further reach a desirable sparsity in $\mathbf{s}_P$, a stronger inhibition β and, potentially, a smaller effective self-excitation $\alpha+\beta$ is necessary. This condition puts a certain amount of pressure on an HD network for which it makes all the other states with less sparsity unstable and thus potentially compromising optimal learnability.

On the other hand, if the amount of orthogonal codewords needed in the sampling process is relatively small—e.g. learning less environments—it might be better to have a non-translationally invariant sampler to produce sparse input $\mathbf{h}_G$ from the beginning such that an HD network can keep its optimal learnability—i.e. $\alpha+\beta=1$ as it was in a balanced initialization. A variant Hebbian rule in the framework of competitive learning has been proposed (Rolls et al., 2006) to learn a non-translationally invariant feedforward projection $U_{DG,G}$ from grid cells to dentate gyrus (DG) for achieving a sparse transformation:

$$\mathbf{h}_G = U_{DG,G}\mathbf{s}_G, \tag{3.23}$$

where $\mathbf{h}_G$ has the desirable sparsity. Importantly, although the learning happens only in the feedforward connections, it involves a WTA network for ac hiving such sparsity too. For this, and for not compromising learnability of the HD network, one needs to introduce an extra layer between the grid cells and place cells—i.e. DG with mutually inhibitory synapses among cells. For our setup, the full system consists of three RNNs that are feedforwardly connected corresponding to MEC-DG-CA3 pathway of the triple system (Witter and Amaral, 2004), where DG is a kWTA network with strong inhibition and CA3 is an HD network. Having said that, for the purpose of focusing on the sequential learning property, we shall first keep eyes on the learning of the recurrent connectivity of CA3 place cells. In the proceeding discussion, I will thus stick with the earlier place-grid dual system with the assumption that $\mathbf{h}_G$ has already a desirable sparsity such that $U_{PG} \equiv \mathbf{I}$ for simplicity.

3.5.2 Online similarity matching with BPTT algorithm

Having the baseline sequential learner ready as a kWTA network, I will then discuss how one can train such a network such that the learning process approximates the

online sparse manifold transformation introduced in Section 3.2.2. I will start by discussing the biological plausibility of using backpropagation through time (BPTT) algorithm, presenting full setup of the RNN learning scheme, and finally discussing how such a scheme also resembles learning a graph analogous to what the optimal learning algorithm does as we discussed in Section 3.2.4

Local BPTT approximates biological synaptic learning with a second-long STDP

In order to properly perform an online similarity matching, the agent has to "remember" all grid cells codewords happened within a short time interval between two adjacent landmarks. To be more precise, it has to remember all codeword correlations between t and $t' \in [t_l, t_{l+1}]$ where the time interval on average is $\Delta t = t_{l+1} - t_l \approx 1s$. This time interval is also the backprop timestep required for BPTT. At first glance, the requirement of such a long synaptic memory does not at all seem plausible given that a cell membrane time constant is as short as 5 to $20ms$. In spite of that, remarkably, a behavioral time scale synaptic plasticity has actually been observed in place cells (Bittner et al., 2017) such that a second-long spike timing dependent plasticity (STDP) rule (Schiller et al., 2018) could be considered as a plausible mechanism in our learning scheme.

However, even knowing the plausibility of a long synaptic memory, it is still important to find an alternative to BPTT given the biological implausibility of such an algorithm (Bengio et al., 2015). Some progress has been made (Scellier and Bengio, 2016; Bellec et al., 2019) in this regard, or regarding learning a long time scale association under only a short plasticity window (Brea et al., 2016). In the context with a similarity matching objective, Pehlevan et al. (2015) and Pehlevan et al. (2018) has shown that backpropagation algorithm can be reduced to hebbian learning for the feedforward connections and anti-hebbian learning for the recurrent connections; after learning, the network establishes a one-to-one mapping between inputs and outputs such that the similarity of the inputs is preserved. In our case, this mapping between inputs and outputs does not exist, so this hebbian-anti-hebbian learning rule needs to be modified to emphasize the learning of recurrent connectivity for an autonomous generation of outputs. Overall, finding an alternative to BPTT is an important future direction; the focus shall be to discover an appropriate architecture that resembles characteristics of the hippocampus such that BPTT can be well approximated by a biologically plausible alternative.

Setup of a candidate RNN learning scheme

I. Forward propagation and random landmark sampling. The architecture is a simple RNN that continually modifies its connectivity across multiple environments as shown in Figure 3.12. The RNN has dynamics of a canonical neural network:

$$\mathbf{s}_P^{t+1} = (1-\eta)\ \mathbf{s}_P^t + \eta\ \left[W_{PP}\mathbf{s}_P^t + b_P + U_v\mathbf{x}_v^t + U_{PG}\mathbf{s}_G^t \times \delta(r(t) - r_l)\right]_+, \tag{3.24}$$

where $\eta \equiv \frac{\Delta t}{\tau} = \frac{2\ ms}{20\ ms} = .1$, $[u]_+ = \begin{cases} 0 & \text{if } u < 0 \\ 1 & \text{if } u \geqslant 0 \end{cases}$, and r_l's are the landmark sampling locations which specify where the grid cell feedforward input is turned on for sampling a place cell codeword $\mathbf{s}_P^t$. The recurrent weight are initialized as $W_{PP} = W_{kWTA}$ and trained subsequently by modifying ΔW from its initial value: $\Delta W = 0$. Note that in order to effectively sample codewords as unbiased as possible (every cell has equal probability to be selected as a k-winner), a detailed study and finetuning on the effect of U_{PG}, U_v, W_{kWTA} to the learning dynamics is required. The interval of one landmark sampling location can also be more flexible than using a delta function, e.g., $\delta(r(t) - r_l) \to e^{-(r(t)-r_l)^2/\sigma_l^2}$.

II. Backpropagation. The two main training objectives are 1) to perform similarity matching between the local codeword correlation matrix of grid cells and that of place cells and 2) to maintain a consist place cell codewords at all landmark sampling locations.

$$L1 = \sum_{|t-t'|<T} \left|\mathbf{s}_P^{\mathrm{T}}(t)\ \mathbf{s}_P(t') - \mathbf{s}_G^{\mathrm{T}}(t)\ \mathbf{s}_G(t')\right|^2, \tag{3.25}$$

$$L2 = \sum_{t=t_0}^{t_0+T} \sum_{l} \delta(r(t) - r_l) \cdot |\mathbf{s}_P(t) - \mathbf{s}_P(t_{l,-1})|^2, \tag{3.26}$$

where $L1$ demands two close-by place cell codewords at t and t'—where $|t - t'| < T$ and $T \approx 1s$ is the backprop time interval—to have the similarity of the corresponding grid cell codewords. The similarity matching loss function $L1$ is "online" because the it is restricted to compare only temporally close-by codewords. $L2$ attempts to make the RNN to consistently produce the same activity patterns at the sampling locations. $L2$ is important for a repeatable landmark sampling because the grid cell activity $\mathbf{s}_G$ is not the only contributor to the sampling process. The recurrent weight W_{PP} has to learn in a way such that the current activity at a landmark location is more or less equal to the last activity at this location $\mathbf{s}_P(t_l) \approx \mathbf{s}_P(t_{l,-1})$. In practice, $L2$ may be

replaced by another self-localization type of loss function—similar to the one used in Section 3.3—for producing stable spatial tuning.

III. Learning an allocentric transformations. This part of learning is not for place field formation, but rather for a robust reactivation on the learnt place fields in a familiar environment.

$$L3 = \sum_{t=t_0}^{t_0+T} \sum_{k,l} \delta(r(t) - r_l) \cdot \left| U_{PG}\mathbf{s}_G(t) - U_{Px^{(k)}}\mathbf{x}^k_{\text{sen}}(t) \right|^2 \tag{3.27}$$

Essentially $U_{Px^{(k)}}$ is trained such that the other sensory inputs of vision, audition, olfaction, etc. can be a substitute—when the grid cell input is unavailable to provide a landmark correction—after learning a new set of place fields: $U_{Px^{(k)}}\mathbf{x}^k_{\text{sen}}(t) = U_{PG}\mathbf{s}_G(t)$. It is important to keep in mind that these learnt feedforward connections are expensive because they are in general not generalizable. In other words, any combinations of these cues happen at a certain location has an exclusive feedforward weight for recreating a desirable landmark input. In doing so, these sensory inputs are only activated at few sparse sampled locations, and the rest of self-localization requires path-integration.

IV. Total loss function. The first two loss functions are major part of the optimization principle and can be minimized together using backpropagation though time algorithm (BPTT)

$$\min_{\Delta W} L1 + L2 \tag{3.28}$$

The third loss function is optimized separately with backpropagation only on the feedforward weights as

$$\min_{U_{Px^{(k)}}} L3 \tag{3.29}$$

V. A necessary forgetting. Besides the above forward-propagation dynamics on activity and backpropagation on weight ΔW, there is a weight decay mechanism necessary for protecting the overall learnability:

$$\tau_W \frac{d\ \Delta W(t)}{dt} = -\Delta W(t), \tag{3.30}$$

where τ_W is the decay time—which could be on the order of hours or even days depending on the specific needs—for connectivity reverting back to an unbiased W_{kWTA}. Note that a more sophisticated weight decay dynamics may be implemented for a better performance (Benna and Fusi, 2016). This crucial active forgetting mechanism

or a more deliberate version has been recently suggested as an important function of the sharp-wave ripple event in hippocampus (Norimoto et al., 2018).

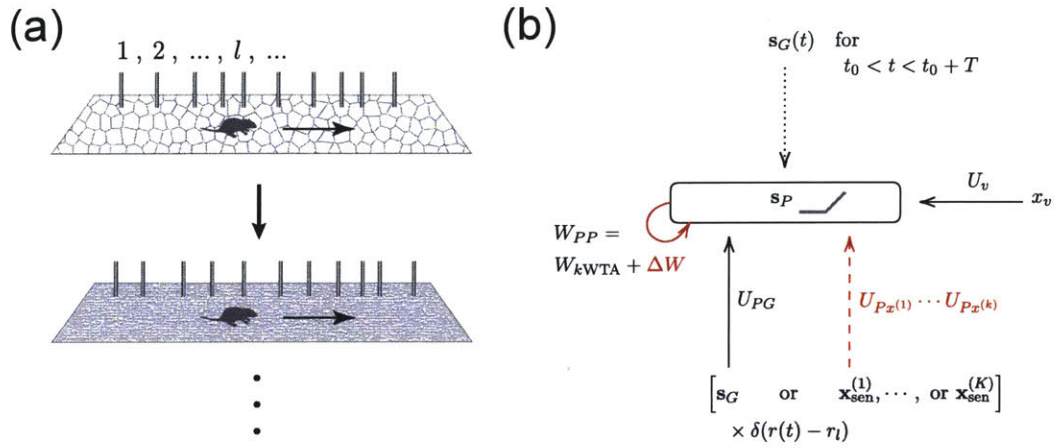

Figure 3.12: A plausible neural implementation of the optimal online learning algorithm. **(a)** The agent is sequentially exposed in multiple environments. The vertical posts indicates the location for optimal random landmark sampling. **(b)** An RNN architecture is used with a trainable recurrent layer that is initialized as a k-winner-take-all network. The two algorithmic steps are implemented as follows. 1) The RNN only receives a grid cell input at landmark locations as an approximation of the *optimal random landmark sampling*. 2) The supervised signal—constructed as a similarity matrix between the grid cell codewords within one second time interval at different times—is used to approximate *online similarity matching*. Finally, the learning of feedforward projections for certain egocentric sensory cues can be done independently such that they can be a substitute of grid cell inputs for error correction at landmark locations. Red arrows indicate trainable connections.

Learning in the optimization principle resembles learning of a graph

For the above learning scheme to work properly, the RNN dynamics has to be high-dimensional as discussed in Section 3.3. The initial energy landscape can be seen as initially flat with a few local minima as shown in Figure 3.13(d). These local minima are the states of sampled landmarks each with a small localized basin. Note that these energy minima are inherently nonautonomous which is only "present" when the corresponding external inputs exist. Over the course of learning, a particular task-relevant trajectory might be repeatedly taken: Figure 3.13(a-c) with specific conjunctions among these landmark sampling locations. Crucially, the structure of these conjunctions—forming a graph—is gradually captured during training: Figure 3.13(d-f). For those paths that have never been taken, e.g. from Location 2 to 4, there will be no stable states assigned to encode these positions along the way. This illustration demonstrates not only that a place cell code is fundamentally trajectory dependent,

but also that it is more topological than topographical (Chen et al., 2014)—in the sense that a place cell code is more of learning a task-specific graph but less of learning a generalizable spatial map (also see Section 3.2.4 for a relevant discussion).

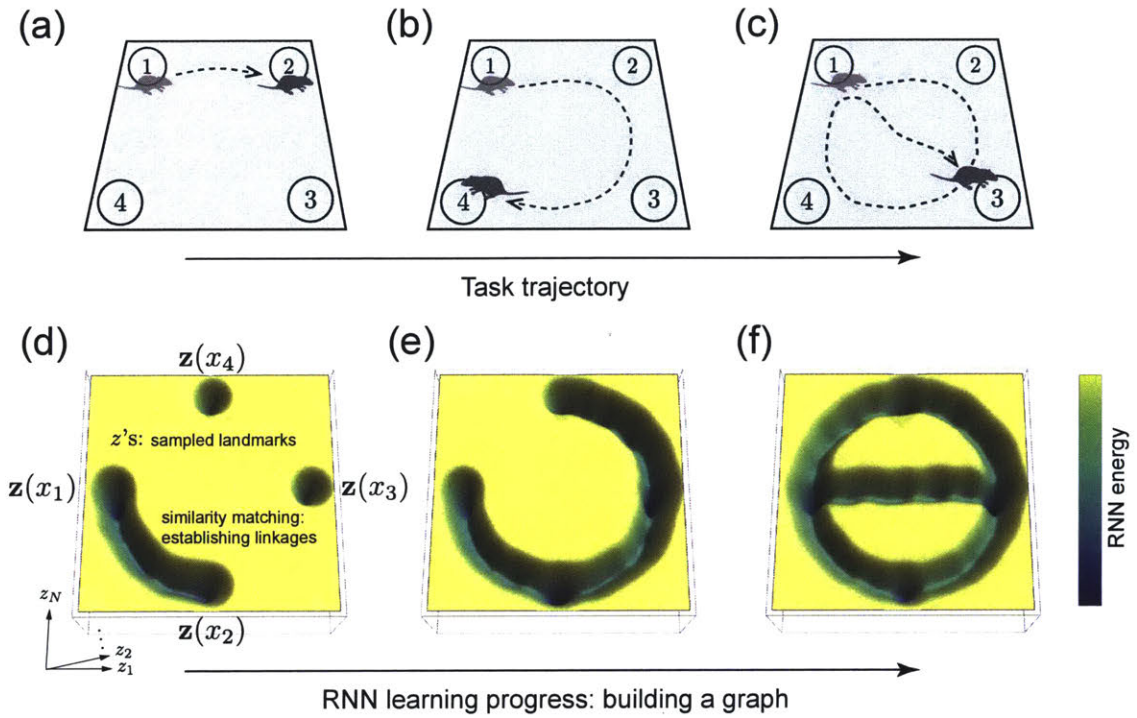

Figure 3.13: A RNN implementation of the optimal online learning algorithm learns the graph that underlies a certain task trajectory. **(a-c)** An example task trajectory that requires the agent to run among four corners without ever walking on the path connecting Location 2 and 4. **(d)** An initially unbiased kWTA network has a flat energy landscape everywhere except the four isolated effective fixed points that represent four corner locations as sampled landmark states. **(e-f)** A series of energy minima start to be developed and thus connect the four landmark states which eventually forms a corresponding graph that underlies the task trajectory. This illustration shows the topological nature of a place cell code in which the geometry, topology, or dimensionality of the learnt manifold is not predetermined by the structure of the environment, but emergent as a behavior progresses.

3.5.3 Persistent biased field propensity: an outcome of metastable attractor dynamics

One important prediction out of the optimization principle is a biased field propensity in a recurrent place cell ensemble. That is, if a place cell has more fields in one environment, the same cell will also have, on average, more fields in the other environments. The dense recurrence of CA3 place cells make them likely to be the candidate source of such a bias. The biased propensity in CA3 is predicted—caused by

metastable attractor dynamics—to be qualitatively similar to a recent experimental discovery (Lee et al., 2019) in CA1 place cells. The subtle differences regarding place field statistics between CA1 and CA3 predicted will be discussed in Section 3.5.4. From the experiments, the field propensity of each cell can persist for months, which suggests a cause from a systematic bias in the circuit rather than by chance. This and an earlier result (Rich et al., 2014) seem to suggest a place cell function that utilizes a biased random landmark sampling—for which some cells are drawn more than others. However, the optimal learning algorithm we have been discussing so far demands an unbiased landmark sampling process that has the best chance to sample an orthogonal codeword. This thus makes a conflict between the theory and experiment. And to resolve it, I ran a simple simulation of an RNN with biased kWTA recurrent connectivity that captures the main dynamical characteristics of an HD network. We will see how such a biased field propensity must happen at the level of RNN implementation.

Setup for the simulation

The architecture is a simple RNN that receives a random input x_l every second that last $.1s$. The RNN has dynamics of a canonical neural network:

$$\mathbf{s}_P^{t+1} = (1-\eta)\ \mathbf{s}_P^t + \eta\ \left[W_{PP}\mathbf{s}_P^t + b_P + U_{Px}\mathbf{x}_l^t\right]_+, \tag{3.31}$$

where $\eta \equiv \frac{\Delta t}{\tau} = \frac{2\ ms}{20\ ms} = .1$, $[u]_+ = \begin{cases} 0 & \text{if } u < 0 \\ 1 & \text{if } u \geqslant 0 \end{cases}$, $U_{Px} \equiv \mathbf{I}$, and $\epsilon \ll 1$ specifies the strength of a perturbation of a biased weight $W(N_{\text{patt}})$—on an otherwise unbiased k-winner-take-all (kWTA) RNN: $W_{PP} = W_{kWTA}$. The biased weight is constructed as

$$W(N_{\text{patt}}) = \sum_{\xi=1}^{N_{\text{patt}}} \mathbf{x}_\xi \mathbf{x}_\xi^T, \tag{3.32}$$

where $\mathbf{x}_\xi$ is a random positive pattern drawn from a gaussian distribution: $\mathbf{x}_\xi = [\mathcal{G}(0,1)]_+$. See Appendix B.7 for other simulation details.

Metastable attractor dynamics causes a biased field propensity

Note that because the perturbation term above, $\epsilon W(N_{\text{patt}})$, is much smaller than the unperturbed weight W_{kWTA}, the overall RNN with W_{PP} does not behave like a usual

stable attractor network with stored patterns $\mathbf{x}_\xi$'s. Instead, W_{PP} behaves mostly like a kWTA network without attractors, yet perturbed to produce a biased propensity for each cell. The simulation results are shown in Figure 3.14. One can see that the stronger the perturbation weight $W(N_{\text{patt}})$, the more biased the cell propensity distribution is. The mechanism of the biased propensity can be understand by the following reasoning:

1. *A biased propensity cannot be explained by a biased feedforward input in the optimization principle*: An input from grid cells is unbiased due to an unbiased feedforward projection (Section 3.5.1) and a uniform average firing rate across grid cells.

2. *Recurrent connectivity between place cells necessarily develop a bias after learning just one environment.* Unlike grid cells, a place cell code is environment-specific, or contextual. To encode any contextual information in the recurrent weight, it is necessary to break the translational invariance of the initial kWTA network, given that the feedforward weight needs to be kept unbiased for maximum likelihood of sampling an orthogonal codeword.

3. *A landmark sampling is biased in a subsequent learning.* After learning one environment, some place cells naturally fire more frequently than the others—because the RNN is no longer translationally invariant. These cells that fires more has higher probability to be selected as a k-winner.

4. *There will be a runaway bias without a weight decay.* Over the course of learning multiple environments, the bias on field propensity accumulates and accelerates with a tendency towards that only a few cells fire all the time while others being silent. This runaway bias in propensity will co-occur with the catastrophic forgetting on the brink of exceeding RNN capacity limit.

5. The biased field propensity can reach an equilibrium with a weight decay. Here we see an additional function of weight decay other than maintaining learnability—i.e., maintaining an unbiased landmark sampling. For a condition with a certain rate of weight learning and weight decay, the bias on the field propensity could reach an equilibrium such that it could have a certain distribution[16].

[16]It's been observed that the CA1 place cells has a gamma-poisson distribution of the probability of number of fields (Rich et al., 2014)

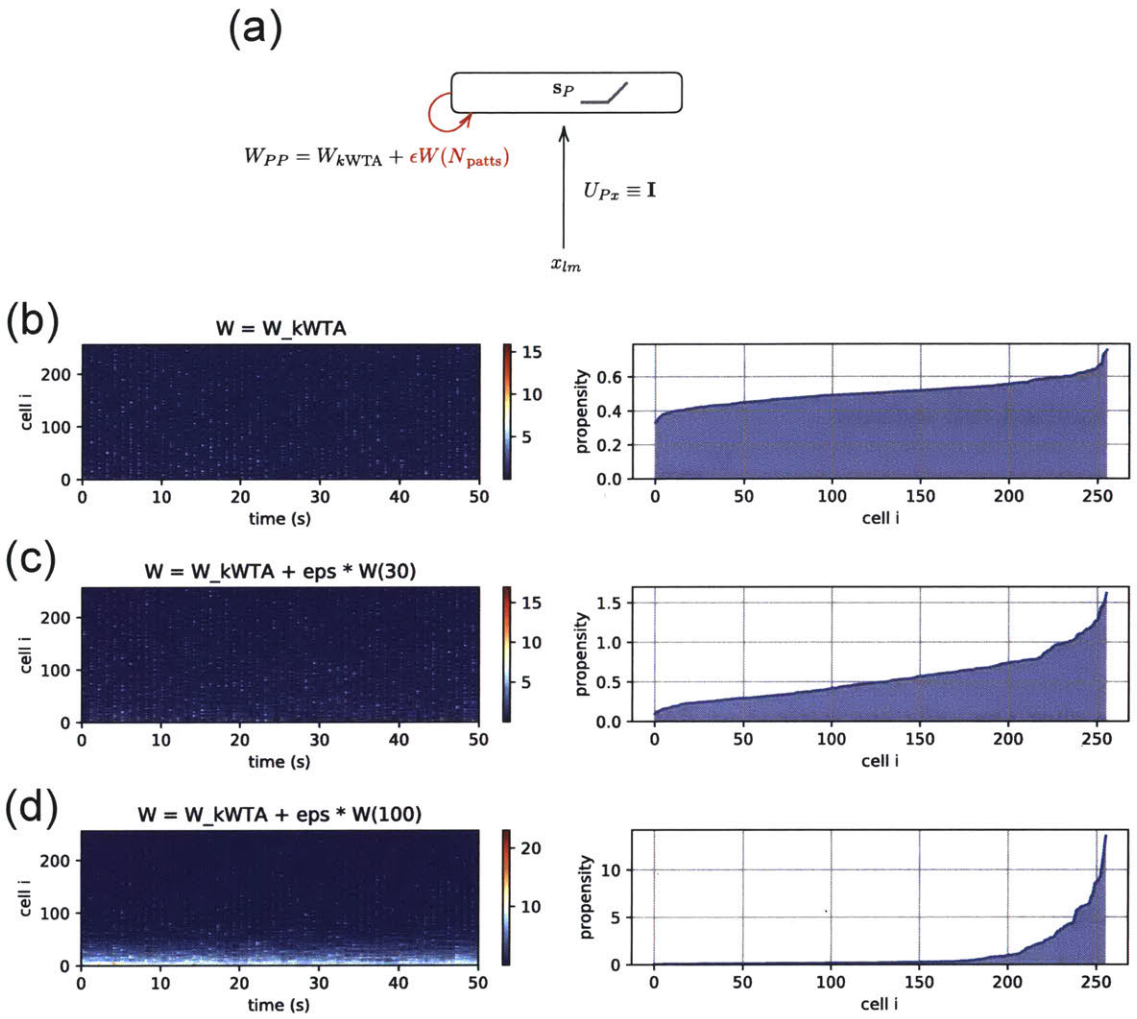

Figure 3.14: The observed persistent field propensity of a place cell ensemble could be a direct consequence of its metastable attractor dynamics during learning. A simplified simulation is used to demonstrate how a weakly biased recurrent weight after learning can result in a biased firing rate under a stream of sparse random inputs (a) A kWTA network with a perturbation in its recurrent connectivity (red) receives unbiased random inputs simulating the optimal random landmark sampling. (b) With an unbiased kWTA network, the average firing rate—or propensity—over 50 seconds is mostly uniform across all cells with only a little bias caused only by undersampling. (c) A larger degree of bias happens when a recurrent weight is perturbed. Note that the perturbation is weak such that no stable attractor is formed. The perturbation causes a biased random landmark sampling (suboptimal) such that some cells are picked as a k-winner more frequently over the course of learning. (d) The bias in recurrent weight will accumulate as learning progresses, and consequently results in a runaway bias in field propensity. It is therefore necessary to have an active forgetting mechanism that revert the recurrent weight towards that of an unbiased kWTA network for protecting the ongoing learnability.

Strong attractor dynamics needs not cause a biased field propensity

From above, one might think that a more conventional continuous attractor network can also cause such a bias in field propensity. So a continuous attractor network might

well be a proper alternative neural implementation that underlies place cells—as long as one can find a way to store multiple maps in such a network like what place cells do. It's actually been demonstrated that such a continuous attractor network exists (Samsonovich and McNaughton, 1997; Battaglia and Treves, 1998; Tsodyks, 2005) named multichart attractor network.

A multichart attractor network stores multiple stable continuous attractors in such a way that the states from different attractors has only little overlaps. A simple example of recurrent weight consists of k identical connectivity profiles but with randomized cell indices:

$$W_{\text{multichart}} = \sum_k W^{(k)}, \tag{3.33}$$

where the matrix elements $w_{i_k,j_k} \in W^{(k)}$ of the k-th maps respect translational symmetry:

$$w_{i_k,j_k} = f(|i_k - j_k|) \tag{3.34}$$

Note that the cells are indiced at different random orders for different environments, e.g. $i_1 = 1, 2, 3, \ldots$ but $i_2 = 5, 3, 7, \ldots$. For getting a single attractor bump, one can choose a simple connectivity profile such that a cell effectively inhibit faraway cells more than its neighbors (Note that these neighbors do not have to be topographically arranged.)

$$f(|i_k - j_k|) = \alpha \exp\left(-|i_k - j_k|^2/2\sigma^2\right) + \beta, \tag{3.35}$$

where $\alpha \geqslant 0$ and $\beta \leqslant 0$. This construction uses all cells for a maps which accumulates the overlaps between attractor states very quickly once more and more maps are stored. A more flexible allocation of resource is to use fraction of cells to construct a map with a blurry spatial boundary between any two maps (Hedrick and Zhang, 2016). In any cases, the multichart continuous attractor can, in principle, reach its maximal capacity if every cells contribute equally in constructing the attractor states. In other words, it is optimal when these cells have an unbiased field propensity for which it does not predict a biased field propensity as observed in the experiment.

On the other hand, this discussion seems to make a multichart attractor network more suitable candidate for neural implementation in the optimization principle than an HD network. However, it is important to keep in mind that a multichart attractor network dynamics is fundamentally low-dimensional, which means that such a network would suffer from exacerbated catastrophic forgetting—as discussed in Section 3.3, and it thereby does possess sequential learnability.

In conclusion, the optimization principle for place cells proposed in this section with a metastable attractor network as its neural implementation naturally predict a biased field propensity. The effect is a direct result from 1) a broken translational invariance in recurrent connectivity after learning and 2) a metastable attractor dynamics that biased an otherwise uniform landmark sampling process. A biased field propensity observed in experiment, therefore, is caused by a dynamical constraint of an RNN implementation rather than a specific functional need.

3.5.4 Experimental predictions in CA1 and CA3 place field statistics

If the biased field propensity is originated from CA3 region (due to its unavoidable biased recurrent circuitry), it is likely that CA1 cells subsequently inherent such a bias given that CA1 place cells receive major inputs from CA3 region (Davoudi and Foster, 2019). From this simple architectural assumption with each CA1 cell receives linear input from multiple CA3 cells, one can predict that

1. a CA1 cell must have on average more place fields, and
2. a CA1 cell must have on average less bias in field propensity.

One should expect that for CA1 place cells to preserve the high spatial information of CA3, each CA1 cell should receive inputs from only a few CA3 place cells coding for various locations. In such a case, CA1 cells will have a biased propensity only slightly smaller than that of CA3.

3.5.5 Remarks and outlook

The neural implementation of the optimization principle given in this section largely realizes the two algorithmic steps in the optimal online learning algorithm introduced in Section 3.2. I demonstrated, using a simplified simulation, how an RNN dynamical constraint inevitably causes a biased field propensity. For future studies, it would be interesting to investigate what kind of distribution of the biased propensity is optimal under such a dynamical constraint. And it would also be interested to further design experiments to see whether an active forgetting mechanism in the brain exists because not only the avoidance of catastrophic forgetting but also the avoidance of biased field propensity for as much as possible.

On the same line of reasoning, there is a sense of *tradeoff between learning speed and memory capacity* in an HD network. A fast learning requires also a fast forgetting to maintain a desirable memory capacity (e.g., remembering k environments). However, fast learning and forgetting could potentially cause, on average, a larger deviation from an unbiased kWTA network, and consequently a larger bias in the field propensity. Because a larger bias would cause effectively a smaller capacity (more cells are potentially silent), more cells are required in an HD network to reach the demanded memory capacity. On the contrary, if one slows down learning, a less biased field propensity is expected so that less cells are needed for reaching the same memory capacity. It would be interesting to design relevant experiments or numerical experiments to test this hypothesis.

Meanwhile, from a theoretical viewpoint, the learning in LD and HD networks represent two polar extremes of a spectrum of different learning dynamics. On the one end, LD dynamics fits the conventional representation-first construction in which the autonomous dynamics dominates with nonautonomous part being merely a perturbation to drive the state transitions. On the other end, HD dynamics do not have any stable representations, and the nonautonomous dynamics dominates as the learning progresses. An LD network starts learning from the strongest attractor stability, whereas an HD network starts from the lowest and always learns just enough stability for the task. It would be an interesting theoretical problem to identify the class of tasks that requires HD learning dynamics, the class requiring LD learning dynamics, or any classes lie in between.

3.6 Chapter summary

- The descriptive function hypothesis I adopted focuses on the two properties of a place cell code: *Place cells exist for having a sequentially-learnable* and *highly-separable path-integrating code.*
- The function hypothesis consists of two functions that are either of a pure coding property: *having a high coding separability within and across environments*, or of a dynamical property: *capable of learning a novel environment with ability to recall learnt ones.* And because of this clean separation, the optimization principle can be broken down into two steps: 1) finding an optimal learning algorithm within the coding-theoretic framework, and 2) finding a neural implementation that can realize this optimal learning algorithm.

- The optimal online learning algorithm that gives rise to place cells is called the *online sparse manifold transformation.* The algorithm consists of two steps: 1) optimal random landmark sampling and 2) online similarity matching. Both require the assistance from an allocentric spatial code for which grid cells fulfill the role. After learning, place cells operate as a *complementary* code independent of grid cells. The algorithm displays a learnability-capacity tradeoff which leads to the third conjecture in the function complementarity: *A place cell circuit can only learn from experience via synaptic plasticity with a compromise in its coding capacity.*

- The optimal online learning algorithm can be directly extended to include the higher-level coding features regarding topological codes and predictive maps—both of which have been hypothesized as functions of place cells. This extension can be seen as a higher-order learning—after the first-order learning prescribed by the two algorithmic steps—that could lead to a set of skewed place fields.

- To implement the optimal online learning algorithm in a neural substrate, the issue of catastrophic forgetting in an RNN needs to be addressed. I discovered that unlike an low-dimensional (LD) network that has a strong attractor dynamics which exacerbates catastrophic forgetting, an high-dimensional (HD) network with a metastable attractor dynamics enables a sequential learnability. The sequential learning mechanism in the HD network reveals a fundamental principle that relates learnability with conjunctive tuning property, which I termed: the *tuning-learnability correspondence.*

- The HD network and its learning dynamics are compared with an earlier CA3 place cell model: the *multichart attractor network model.* The implausibility and remaining issues for modeling CA3 place cells as a standalone path-integrator is resolved. And experimental implications and predictions are discussed including aspects of *1) stable place fields during awake, 2) cofiring structure during sleep, 3) conjunctive tuning, and 4) global remapping.*

- Having the issue regarding catastrophic forgetting been resolved, I proposed an RNN implementation for the optimal online learning algorithm which analogously implements the two algorithmic steps utilizing the advantage of the HD network dynamics. I discussed how this neural implementation is impossible to be a perfect match, especially in the step of optimal random landmark sampling. For which, I ran a simplified simulation to demonstrate this deviation

(from the optimal algorithm) causes a bias in the landmark sampling process which subsequently causes a biased field propensity. The biased field propensity has been observed in CA1 place cells. I predicted that this bias originates in CA3 place cells as a direct consequence of an unavoidable biased firing in a metastable attractor dynamics instead of a specific functional need.

Chapter 4

Conclusion

This work attempts to understand what functions the brain needs to perform that ultimately lead to the emergence of grid and place cells. In contrast to earlier theoretical frameworks which mostly focused on either modeling phenomena or demonstrating how these cells can perform a specific function, the optimization principle framework I laid out sought for the emergence of grid or place cells as the optimal solution of hypothesized functions. Given that grid and place cells behave qualitatively differently in both their coding and dynamical properties, It is necessary to start—in the optimization principle framework—by assuming these two types of cells perform different sets of functions that are complementary to each other such that grid and place cells are both optimal solutions in their own domains. The specific architecture choice, i.e., the grid-assist architecture, has two systems operating largely independently so that their optimization principles can be separately pursued.

To pursue optimization principle for grid cells, I hypothesized that the functions are of two coding properties: of high-capacity and of path-integration. To start, I trained a recurrent neural network (RNN) to path-integrate in a large environment. Although the RNN failed to converge to a desirable performance, the training schemes bought an invaluable insight that a grid cell ensemble could be a rare island circuit solution that are generally unreachable via incrementally changing synaptic connections. It further implied that a grid cell circuit could be a product from some developmental process rather than learnt from experience via Hebbian rule. To proceed after RNN training schemes, I adopted a pure coding theoretic approach that directly searched a solution space of tuning curves. This approach is justified provided that the earlier works have worked out the circuit solution for an arbitrary set

of grid cell tuning curves, and a grid cell circuit is generally rigid.

The coding theoretic approach in a binary neuron scheme revealed that the path-integration function of grid cells can also be directly evaluated using mere tuning curves, and the objective is to have a translationally invariant (TI) code. However, after generalizing the TI coding property to a continuous neuron scheme. It became clear that an additional biological constraint is necessary for the emergence of grid cells given that the original two functional constraints only led to aperiodic tuning curves. I then used Scheme A to explore a biological constraint that targeted a dense periodic code, and to show that it is necessary to have both an unlikely extra linear denoiser and an implausibly complex capacity measure for the emergence. To resolve these issues, Scheme B, instead, used a biological constraint that targeted a sparse periodic code with only a simple capacity measure—i.e., the coding separability—without the need for a denoiser. However, the optimal solutions in such a scheme turned out to have non-modular and semi-periodic tuning curves. These results forced me to drop the original assumption on using a simple capacity measure that is based on just the average Euclidean distance, and to consider a new capacity measure which accounts for a robust long coding line with only a sufficient noise resistance. In Scheme C with the new capacity measure, I demonstrated, using a simplified six-neuron code, that the energy landscape of the optimization problem is extremely rough as well as a preliminary run showing that all optimized solutions were still far away from the global optimum. The Scheme C is thus the closest to the optimization principle of grid cells before it is proved or disproved by showing whether or not its optimal solution is a grid cell code.

The pursuit of the optimization principle for place cells has the focus opposite to that of grid cells. Unlike grid cells, a place cell circuit is plastic and can change through experience. The challenge, therefore, lies in finding a plausible neural implementation for a relatively straightforward online learning algorithm—which can be shown as the optimal solution at the algorithmic level. To implement such an algorithm, I used an RNN with a plastic recurrent connectivity; an RNN is required since place cells have to perform a different path-integration independently of grid cells under the assumption of their function complementarity. The algorithm consists of two interleaving steps which either sample or interpolate to form place cell tuning curves online. However, for an RNN to perform such two algorithmic steps, a well-known issue of catastrophic forgetting in a sequential learning process needs to be resolved. For that, I investigated how network dynamics might influence the learning process using a simplified spatial

task that demanded an RNN to predict upcoming landmarks. I discovered that while a strong attractor dynamics exacerbates catastrophic forgetting, a metastable attractor dynamics enables sequential learnability as well as producing all major place cell tuning properties. These results resolve the issues of a previous place cell model called the multichart attractor network model—which failed to predict many place cell characteristics with issues involving an implausible prewiring, a need for a central control switch, and a lack of recent-experience reactivation. Moreover, the results from an RNN with metastable attractor dynamics reveals a fundamental principle which I termed the tuning-learnability correspondence. This principle subsequently predicts that conjunctive tuning curves must happen even when a task demands only position coding. Contrasting to my earlier RNN training results for grid cells, the success of the RNN in learning multiple smaller environments suggests another fundamental principle: the capacity-learnability tradeoff. It implies that a grid cell ensemble compromises its plasticity to gain a high coding capacity, whereas a place cell ensemble compromises its coding capacity in exchange for an online learnability. These results clearly showed a complementary coding nature for the dual place-grid cell system.

Having the issue of catastrophic forgetting finally resolved, I laid out a plausible neural implementation that analogously implements the two steps in the optimal online learning algorithm. I illustrated how by applying the sampling and interpolation steps alternately, an RNN could carve a graph in its energy landscape which resembles the essence of topological nature of a place cell code. In the end, I demonstrated and predicted, based on this implementation and the results of metastable attractor dynamics, that a biased field propensity discovered in CA1 place cells could have originated from CA3 as a direct consequence of an inevitable biased landmark sampling process in an RNN that otherwise does not exist at the algorithmic level.

In this thesis, I have demonstrated how the optimization principle framework can guide a rigorous theoretical study that bridges the gap between circuits and functions. I have shown that such a framework can adopt different approaches including various RNN training or coding-theoretic optimization schemes depending on the nature of a targeted system. The fact that an optimization principle demands a plausible neural implementation forces one to find a solution that is simultaneously optimal in both algorithmic and implementation level. As a result, the potential conflict between the two levels could bring new insights to a studied system. As I discovered—based on the fact that place cells exhibit metastable attractor dynamics—that the phenomenon of

remapping, conjunctive tuning, or biased field propensity could be, to a large extent, a result of place cells fulfilling an overall common objective subjected to certain dynamical constraints of a neural network rather than the need for some other specific functions.

Appendix A

Supplementary Information for Chapter 2

A.1 Proof: a TI code can only be generated by incremental rotations

Definition. A state matrix Z is constructed by sequence of column vectors (states), $\mathbf{z}_x$, such that

$$Z = [\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_{L-1}], \tag{A.1}$$

where $x = 1, 2, ..., L-1$ labels L states in N dimensional spaces.

Assumption 1. The initial state $\mathbf{z}_0$ is advanced by a constant linear transformation W resulting in consecutive states:

$$\mathbf{z}_x = W^x \ \mathbf{z}_0. \tag{A.2}$$

Importantly, W is parameterized to be arbitrarily close to identity matrix I.

Assumption 2. The correlation (and therefore Euclidean distance: $d(x, x') \equiv 2 - 2\ c(x, x')$) between any two states is translationally invariant:

$$\begin{aligned} c(x, x') &\equiv \mathbf{z}_x^T \ \mathbf{z}_{x'} \\ &= c(x - x') \equiv \mathbf{z}_0^T \ W^{x'-x} \ \mathbf{z}_0 \end{aligned} \tag{A.3}$$

Implication. W must be an orthogonal matrix such that

$$W^T W = I \tag{A.4}$$

Proof

$$c(x, x') = \mathbf{z}_x^T \, \mathbf{z}_{x'} = \mathbf{z}_x^T \, (W^x)^T \, W^{x'} \, \mathbf{z}_{x'} \tag{A.5}$$

If Assumption 2 holds, i.e. $c(x, x') = c(x - x')$, then

$$z_0^T (W^x)^T W^{x'} z_0 = z_0^T W^{x'-x} z_0, \tag{A.6}$$

which implies

$$(W^x)^T W^{x'} = W^{-x} W^{x'}, \tag{A.7}$$

or equivalently

$$(W^x)^T = (W^x)^{-1}. \tag{A.8}$$

The above shows that W^x is an orthogonal matrix. To show W is also an orthogonal matrix. We use the fact that an arbitrary orthogonal matrix A can be decomposed as $A = PBP^T$, where P is an orthogonal matrix, and B is a block-diagonal matrix:

$$B = \begin{bmatrix} R_1 & & & & \\ & R_2 & & 0 & \\ & & \ddots & & \\ & & & J_1 & \\ & 0 & & & J_2 \\ & & & & & \ddots \end{bmatrix}, \tag{A.9}$$

where R_i is a 2D rotational matrix:

$$R_i = \begin{pmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{pmatrix}, \tag{A.10}$$

and J_i is a reflection matrix:

$$J_i = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{A.11}$$

For our purposes, a path-integrating code has smooth codeword transition such that B should contains only rotational matrices R_i with small rotation angle $\theta_i \to 0$ as

stated in Assumption 1. The reflection blocks J_i can be then eliminated. And the corresponding matrix W^x is a subgroup called special orthogonal group $SO(N)$ out of all possible orthogonal matrices. Assumption 1 therefore leads to a block-diagonal transfomation matrix W^x:

$$W^x \equiv P \begin{bmatrix} R_1 & & & \\ & R_2 & & \\ & & \ddots & \\ & & & R_{N/2} \end{bmatrix} P^T, \text{ if } N \text{ is even,} \tag{A.12}$$

or

$$W^x \equiv P \begin{bmatrix} R_1 & & & & \\ & R_2 & & & \\ & & \ddots & & \\ & & & R_{\text{floor}(N/2)} & \\ & & & & 1 \end{bmatrix} P^T, \quad \text{if } N \text{ is odd.} \tag{A.13}$$

W^x can thus be further diagonalized with a constant unitary matrix U such that

$$W^x = P\ U\ D\ U^\dagger\ P^T, \tag{A.14}$$

where

$$D = \begin{bmatrix} e^{i\theta_1} & & & & & \\ & e^{-i\theta_1} & & & & \\ & & \ddots & & & \\ & & & e^{i\theta_{N/2}} & & \\ & & & & e^{-i\theta_{N/2}} \end{bmatrix} \text{ and} \tag{A.15}$$

$$U = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & & & \\ i & -i & & & \\ & & \ddots & & \\ & & & 1 & 1 \\ & & & i & -i \end{bmatrix} \text{ if } N \text{ is even,} \tag{A.16}$$

or

$$D = \begin{bmatrix} e^{i\theta_1} & & & & & & \\ & e^{-i\theta_1} & & & & & \\ & & \ddots & & & & \\ & & & e^{i\theta_{N/2}} & & & \\ & & & & e^{-i\theta_{N/2}} & & \\ & & & & & 1 \end{bmatrix} \quad \text{and} \tag{A.17}$$

$$U = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & & & & & \\ i & -i & & & & & \\ & & \ddots & & & & \\ & & & 1 & 1 & & \\ & & & i & -i & & \\ & & & & & 1 \end{bmatrix} \quad \text{if } N \text{ is odd.} \tag{A.18}$$

Note that as long as the matrix W^x can be diagonalized to the above form with D containing unity complex conjugate pairs, W^x belongs to special orthogonal group. We can then use this result to simplify expression of W^x:

$$W^x = P\ U\ D\ U^\dagger\ P^T = P\ U\ E^x\ U^\dagger\ P^T = (P\ U\ E\ U^\dagger\ P^T)^x, \tag{A.19}$$

where $E^x \equiv D$. Thus we have

$$W = P\ U\ E\ U^\dagger\ P^T \tag{A.20}$$

with

$$E = \begin{bmatrix} e^{i\theta_1/x} & & & & \\ & e^{-i\theta_1/x} & & & \\ & & \ddots & & \\ & & & e^{i\theta_{N/2}/x} & \\ & & & & e^{-i\theta_{N/2}/x} \end{bmatrix} \quad \text{if } N \text{ is even,} \tag{A.21}$$

or

$$E = \begin{bmatrix} e^{i\theta_1/x} & & & & & \\ & e^{-i\theta_1/x} & & & & \\ & & \ddots & & & \\ & & & e^{i\theta_{N/2}/x} & & \\ & & & & e^{-i\theta_{N/2}/x} & \\ & & & & & 1 \end{bmatrix} \text{ if } N \text{ is odd.} \tag{A.22}$$

Since E also has the form of unity complex conjugate pairs in the diagonal just like D, W also belongs to special orthogonal group just like W^x. At last, we have proved that a TI code under the two assumptions can only be generated by a orthogonal transformation.

A.2 A typical continuous TI code is not strictly positive

Although strictly positive TI codes exist, they are not typical. Below is a simple 4-cell example.

$$z(x) = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos x \\ \sin x \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1+\cos x \\ 1+\sin x \\ 1-\cos x \\ 1-\sin x \end{pmatrix} \tag{A.23}$$

Positive TI codes are islands in a much larger group of partially positive TI codes. An example of partially positive tuning curve out of a TI code is shown in Figure A.1. Since a positive TI code is an island in solution space, A strictly nonnegativity constraint risks a gradient-based optimization to be trapped in a local minimum of the initial island.

It is for this reason, I used soft nonnegativity constraint in Scheme B for the loss function, $L3$. The definition of nonnegativity does not completely eliminate negative portion of tuning curves, it therefore encompasses more potential solutions that has sparse large positive rate but dense small negative rate, similar to orange curves in Figure A.1.

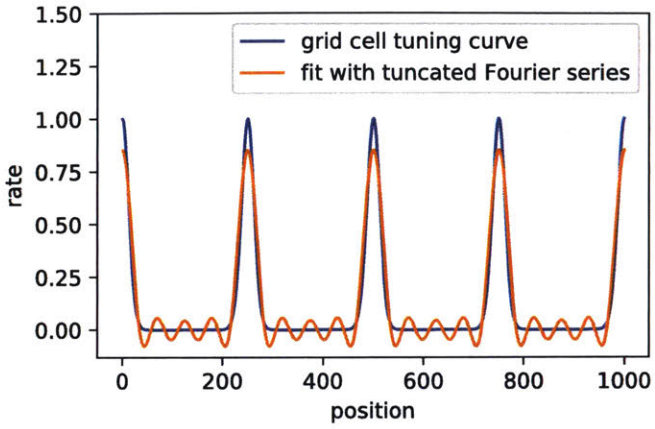

Figure A.1: A typical continuous TI code is not strictly positive. In coding theoretic approach, a fixed maximal frequency is used—in Fourier basis functions—to implicitly set an intrinsic noise. A tuning curve, therefore, is a linear sum of a truncated Fourier series.

A.3 Counting modules

For Scheme A-1, the target tuning curves are of single frequency, so that we can count modules directly from spectrum.

For Scheme A-2, the peaks in the spectrum are no longer single frequencies. In this scheme, I first used Gaussian smoothing with $\sigma = 1/\texttt{num\_freqs}$ (See Appendix A.13 for specific values used in the simulations). Second, I set the spectral density value smaller than .2 to zero. Finally, I count the remaining maxima with local curvature larger than 10^{-4}.

For Scheme B-2, the target tuning curves are not simple sinusoidal, one cannot count modules from spectrum. Instead, I measure the similarity between two tuning curves to decide whether they belong to the same module. The similarity is defined as the maximum of cross circular convolution function:

$$Q_{ij} = \frac{N}{L} \max_x \left(\sum_{x'} z_i(x) z_j(x - x') \right) \tag{A.24}$$

The threshold of $Q_{ij} > .7$ is chosen for grouping z_i and z_j to the same module. To automatically counting modules from Q_{ij}. I interpret Q as an adjacency matrix for a corresponding graph $\mathcal{G}$. At last I used standard Depth First Search to count disconnected components in $\mathcal{G}$. I.e.

$$\textit{number of modules} \;=\; \textit{number of disconnected components in } \mathcal{G} \tag{A.25}$$

A.4 Analytical capacity upper-bound for unimodal tuning curves

An approximate upper-bound for separability can be analytically derived given that there are enough cells for a code with unimodal tuning curves to be an optimal solution. In this derivation, I ignored the boundary effect of finite coding range. The tuning curves of N cells are assumed to be gaussian function:

$$z_i(x) \equiv \frac{1}{\pi^{1/4}\sqrt{\sigma}} \exp\left(-\frac{(x-x_i)^2}{2\sigma^2}\right) \tag{A.26}$$

The correlation matrix of a pair of codewords, $z_i(x)$ and $z_i(x')$, is given as

$$\begin{aligned} c(x,x') &\equiv \int_{-\infty}^{\infty} z_i(x)\ z_i(x')\ dx_i \\ &= \exp\left(-\frac{(x-x')^2}{4\sigma^2}\right). \end{aligned} \tag{A.27}$$

The corresponding distance matrix is then

$$\begin{aligned} d(x,x') &\equiv \sqrt{\int_{-\infty}^{\infty} (z_i(x) - z_i(x'))^2\ dx_i} \\ &= \sqrt{2 - 2\ c(x,x')} \\ &= \sqrt{2 - 2\ \exp\left(-\frac{(x-x')^2}{4\sigma^2}\right)}. \end{aligned} \tag{A.28}$$

The last expression can be well approximated by a tanh function:

$$d(x,x') = \sqrt{2 - 2\ \exp\left(-\frac{(x-x')^2}{4\sigma^2}\right)} \approx \sqrt{2}\ \tanh\left(\frac{|x-x'|}{2\sigma}\right). \tag{A.29}$$

Separability defined as the average of distance matrix is therefore analytically trackable:

$$\begin{aligned} S &\equiv \frac{1}{L}\int_0^L d(x,x')\ d|x-x'| \\ &= \frac{1}{L}\int_0^L \sqrt{2}\ \tanh\left(\frac{|x-x'|}{2\sigma}\right)\ d|x-x'| \\ &= \sqrt{2}\ \left(1 - \log 4 \cdot \frac{\sigma}{L}\right). \end{aligned} \tag{A.30}$$

A.5 Stochastic orthogonal gradient descent

In the stochastic orthogonal gradient descent (SOGD), the update of the coefficient matrix P is based on a single loss function at a time. Loss function ℓ is chosen at random at epoch t to compute a gradient that is orthogonal to all other gradients based on loss functions $k \neq \ell$:

$$\Delta P^t(\ell) = \nabla_\ell - \sum_{k \neq \ell} \frac{\nabla_\ell \cdot \nabla_k}{\nabla_k \cdot \nabla_k} \nabla_k, \tag{A.31}$$

where $\nabla_\ell \equiv \nabla_P$ Loss ℓ. The probability of choosing certain loss function is specified as $[p_1, ..., p_K]$ for all K loss functions in total. The update to P is given as

$$P_{ij}^{t+1} = P_{ij}^t + \Delta P_{ij}^t(\ell) \cdot lr(\ell) \cdot M_{ij}(\alpha_\ell), \tag{A.32}$$

where $lr(\ell)$ is relative learning rate for Loss ℓ and $M_{i:}(\alpha_\ell) = 1$ or 0 is a mask that update only α_ℓ cells at a time.

A.6 Modular solutions must be island solutions to be optima

Figure A.2 uses a three-cell two-module solution as an example to illustrate how a smooth transition through a series of high-capacity two-module solutions do not exist. The two modular solutions are topologically non-equivalent in a sense that the transition between them must go through a different class (similar to genus in topology), i.e., a non-modular solution.

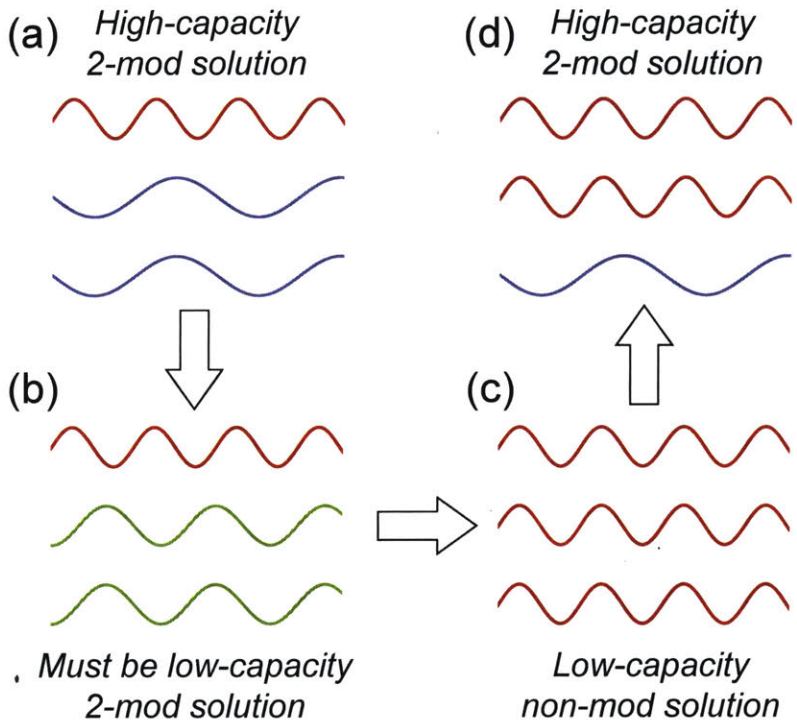

Figure A.2: Any two high-capacity modular solutions are separated by low-capacity non-modular solutions. Assuming (a) and (d) are two high-capacity 2-module solutions, a smooth transformation without breaking the number of modules is performed by increasing the frequency of one of the modules in (b) until all three cells have the same frequency in (c) before the bottom cell is allowed to decrease its frequency again to match the solution in (a). This invariant transformation must go through a non-modular solution in (c), which is a low-capacity solution by definition.

A.7 New capacity measure is a piecewise discrete function of tuning curves

In Figure 2.23(f), one can see that the definition of new capacity results in a piecewise discrete function. Figure A.3 below is a pictorial explanation on the origin of this discreteness of new capacity.

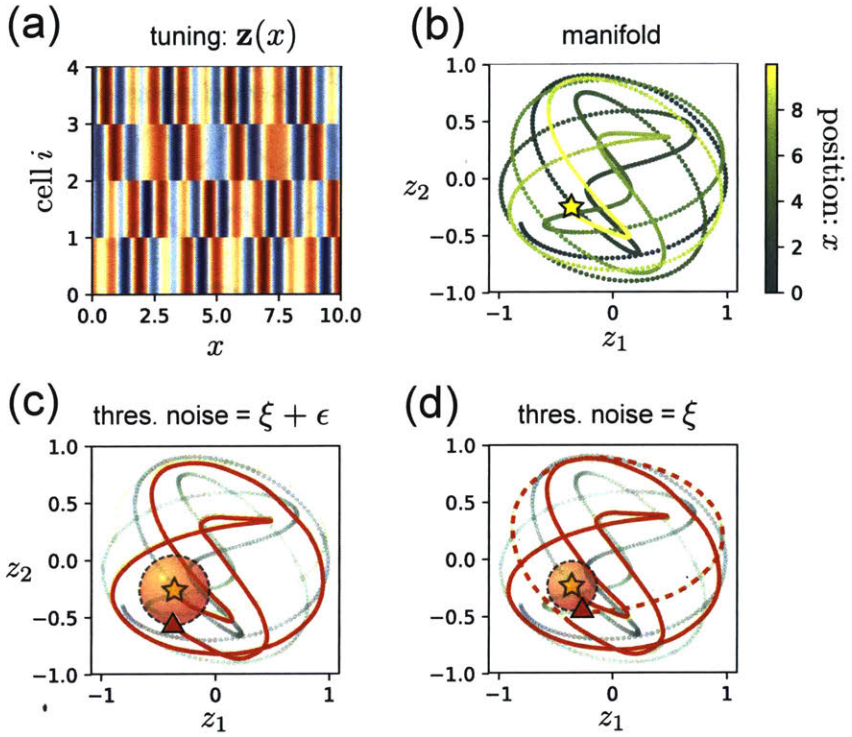

Figure A.3: New capacity exhibit discrete jump in value when sweeping the threshold noise. **(a)** Tuning curves of a high-capacity 4-cell TI code with frequencies: $(f_1, f_2) = (1, 1/\varphi)$. **(b)** Coding line (1D manifold) of tuning curves of (a). **(c)** The length of red segment of the coding line from the yellow star to the red triangle (first collision point with the noise ball) is defined as the capacity under threshold noise $\xi + \epsilon$ as the radius of the noise ball. **(d)** Decreasing the threshold noise by a small amount results in a discrete increasing in capacity as depicted by the dashed red segment.

A.8 RNN training hyperparameters for Scheme 1

Table A.1: Scheme 1 — Place cells with WTA dynamics

name	codename	dim.	value	description
N_P	num_pc		2000	
N_G	num_gc		200	
W_{PP}	Wpp	$N_P \times N_P$	$(\alpha_P + \beta_P)\,\mathbf{I} - \beta_P$, where $\alpha_P = .8$ and $\beta_P = .4/N_P$	WTA RNN
W^0_{GG}	Wgg	$N_G \times N_G$	$(\alpha_G + \beta_G)\,\mathbf{I} - \beta_G$, where $\alpha_G = .8$ and $\beta_G = 0$	near identity initialization
W^0_{GP}	Wgp	$N_G \times N_P$	$\mathcal{G}(0, .1/N_P)$	random initialization
W^0_{PG}	Wpg	$N_P \times N_G$	$\mathcal{G}(0, 1/N_P)$	random initialization
b_P	bp		0	const. bias to PCs
b_G	bg		.2	const. bias to GCs
a_v	cue_strength		.1	
a_l	vel_strength		.1	
f	frac		.3	aim to set fraction of active PCs
U_v	Uv	$N_G \times 1$	U_v [odd] $= a_v$ linspace$(1, .5, N_G/2)$; U_v [even] $= -a_v$ linspace$(1, .5, N_G/2)$	vel. input weights to GCs
U_l	Ul	$N_P \times N_l$	fN_P nonzero entries sampled via $\mathcal{G}(a_l\ , .5\ a_l\ )$ truncated at lower & upper bounds $= .5\ a_l$ and $1.5\ a_l$	landmark input weights to PCs
η	eta		.1 ($\Delta t/\tau$ = 2.5ms / 25ms)	Δt: discretized time-interval τ: membrane time constant
a_{rand}	a_rand		.0002	random acceleration strength
v_{max}	v_max		.01 (.1cm/2.5ms = 40cm/s)	max. vel.
a	a		$a^{t+1} = \mathcal{G}(0, a_{rand})$	acceleration
x_v	x[1,t]		$v^{t+1} = \text{clip}\left(v^t + a^t, -v_{max}, v_{max}\right)$	velocity inputs
r	x[0,t]		$r^{t+1} = \left(r^t + v^t\right) \bmod L$	position on a ring
L	env_size		500 (50m)	

Continued on next page

Table A.1 – continued from previous page

name	codename	dim.	value	description
N_l	num_lm		50	number of landmarks
σ_l	lm_width		1	
r^i	lm_pos		[0, 10, 20, 27, 42, 51, 60, 70, 77, 92, 103, 110, 119, 129, 139, 147, 163, 171, 177, 189, 198, 213, 222, 227, 238, 253, 258, 269, 283, 293, 297, 309, 322, 327, 341, 353, 359, 373, 381, 391, 397, 408, 419, 428, 443, 448, 460, 470, 478, 488]	
x_l^i	x[2+i,t]		$x_l^{i,t} = \exp\left[-\left(x^t - r^i\right)^2 / \sigma_l^2\right]$	landmark inputs
	batch_size		10	
	num_batches		10	
T	timesteps		500 (500 * 2.5ms = 1.25s)	backprop timesteps
lr	learning_rate		.0001	

A.9 RNN training hyperparameters for Scheme 2

Table A.2: Scheme 2 – Softmax nonlinearity to approximate WTA dynamics

name	codename	dim.	value	description
N_P	num_pc		1000	
N_G	num_gc		100	
W_{GG}^0	Wgg	$N_G \times N_G$	$(\alpha_G + \beta_G)\,\mathbf{I} - \beta_G$, where $\alpha_G = .8$ and $\beta_G = 0$	near identity initialization
W_{PG}^0	Wpg	$N_P \times N_G$	$\mathcal{G}(0, 20/N_P)$	random initialization
b_G	bg		.2	const. bias to GCs
a_v	cue_strength		.05	
a_l	vel_strength		.1	
f	frac		.1	aim to set fraction of active PCs

Continued on next page

Table A.2 – continued from previous page

name	codename	dim.	value	description
U_v	`Uv`	$N_G \times 1$	U_v [odd] = a_v linspace$(1, .5, N_G/2)$; U_v [even] = $-a_v$ linspace$(1, .5, N_G/2)$	vel. input weights to GCs
U_l	`Ul`	$N_P \times N_l$	fN_P nonzero entries sampled via $\mathcal{G}(a_l, .5\ a_l)$ truncated at lower & upper bounds = .5 a_l & 1.5 a_l	landmark input weights to PCs
U_{Gl}	`Ulgc`	$N_G \times N_l$	uniform$(-1, 1)$	landmark input weights to GCs
η	`eta`		.1 ($\Delta t/\tau$ = 2.5ms / 25ms)	Δt: discretized time-interval τ: membrane time constant
a_{rand}	`a_rand`		.0002	random acceleration strength
v_{max}	`v_max`		.01 (.1cm/2.5ms = 40cm/s)	max. vel.
a	`a`		$a^{t+1} = \mathcal{G}(0, a_{rand})$	acceleration
x_v	`x[1,t]`		v^{t+1} = clip $\left(v^t + a^t, -v_{max}, v_{max}\right)$	velocity inputs
r	`x[0,t]`		$r^{t+1} = \left(r^t + v^t\right) \bmod L$	position on a ring
L	`env_size`		500 (50m)	
N_l	`num_lm`		50	number of landmarks
σ_l	`lm_width`		1	
r^i	`lm_pos`		[0, 10, 21, 30, 39, 47, 59, 65, 72, 86, 96, 107, 117, 124, 133, 145, 158, 168, 176, 190, 200, 209, 213, 226, 238, 251, 258, 271, 280, 286, 295, 301, 307, 316, 327, 343, 355, 363, 371, 381, 389, 408, 415, 419, 429, 440, 452, 461, 471, 486]	
x_l^i	`x[2+i,t]`		$x_l^{i,t} = \exp\left[-\left(x^t - r^i\right)^2 / \sigma_l^2\right]$	landmark inputs
	`batch_size`		10	
	`num_batches`		10	

Continued on next page

Table A.2 – continued from previous page

name	codename	dim.	value	description
T	`timesteps`		500 (500 * 2.5ms = 1.25s)	backprop timesteps
lr	`learning_rate`		.01	

A.10 Optimization hyperparameters for Scheme A-1

Table A.3: Scheme A-1 – Ad hoc denoising model

name	codename	value	description
L	`num_periods`	40	coding range
	`num_points`	15	number of discrete points in a period
U	`num_freqs`	280	number of frequencies
V	`num_phis`	8	number of phases
N_b	`num_basis`	UV	number of basis functions
N	`num_cells`	UV	number of cells
a_{var}	`a_var`	0, .2, .4, .6, .8, 1	
a_j^0	`init_a[j]`	$\mathcal{G}(1, \sqrt{a_{var}}) \times$ RandSgn	first part of the coefficient matrix
P_{ij}^0	`init_P[i,j]`	$\mathcal{G}(0,1)$	second part of the coefficient matrix
β	`bn`	0,...,10 and $\sqrt{N}$	ad hoc correlated noise level
$lr0 \sim 3$	`lr0, lr1, lr2, lr3`	5, 1, 1, 1/12	baseline and relative learning rates for three loss functions
lrA, lrP	`lrA, lrP`	5e-6, 1	relative learning rate for P_{ij} and a_j
T	`num_epochs`	500	total training epochs
	`sch3`	$\left[1 + \exp\left(-\frac{t - T/3}{T/40}\right)\right]^{-1}$	$lr3$ is scheduled to be on after $T/3$

A.11 Optimization hyperparameters for Scheme A-2

Table A.4: Scheme A-2 – Denoising model that implements correct tradeoff

name	codename	value	description
L	num_periods	40	coding range
	num_points	15	number of discrete points in a period
U	num_freqs	280	number of frequencies
V	num_phis	2	number of phases
N_b	num_basis	UV	number of basis functions
N	num_cells	UV	number of cells
a_{var}	a_var	.6	
a_j^0	init_a[j]	$\mathcal{G}\left(1, \sqrt{a_{var}}\right)$	only used to initialize P
p_{ij}^0	init_p[i,j]	$\mathcal{G}(0,1)$	only used to initialize P
P_{ij}^0	init_P[i,j]	$p_{ij}^0\ a_j^0$	coefficient matrix
δ	bn	1~20	threshold noise level
$lr0 \sim 3$	lr0, lr1, lr2, lr3	5, 1, 1, 1/30	baseline and relative learning rates for three loss functions
lrP	lrP	1	relative learning rate for P_{ij} and a_j
T	num_epochs	5000	total training epochs
ΔT	d_epoch	500	
	sch1	$\begin{cases} 0 & \text{if } t \bmod \Delta T < \Delta T/2 \\ 1 & \text{if } t \bmod \Delta T \geqslant \Delta T/2 \end{cases}$	alternating schedule for $lr1$ and $lr2$
	sch3	$\left[1 + \exp\left(-\frac{t - T/2}{T/100}\right)\right]^{-1} \cdot sch1(t)$	alternating schedule for $lr3$

A.12 Optimization hyperparameters for Scheme B-1

Table A.5: Scheme B-1 – Preliminary of Scheme B-2

name	codename	value	description
L	num_periods	20	coding range
	num_points	10	number of discrete points in a period

Continued on next page

Table A.5 – continued from previous page

name	codename	value	description
U	num_freqs	200	number of frequencies
V	num_phis	2	number of phases
N_b	num_basis	UV	number of basis functions
N	num_cells	10~80	number of cells
P_{ij}^0	init_P[i,j]	$\mathcal{G}(0,1)$	coefficient matrix
$lr0$, $lr1$, $lr3$	lr0, lr1, lr3	.01, .01, 1	baseline and relative learning rates for loss functions
$r_{z,tar}^2$	r2z1	.001	target tuning negativity
w	w	.5~4	intrinsic noise to set maximal frequency $1/w$
T	num_epochs	1e5	total training epochs
β	beta	.1	last learning rate in annealing schedule
	sch	$\left[1 + \frac{1}{\beta - 1}\left(\frac{t - T/4}{T - T/4}\right)^2\right]^{-1}$	$\frac{1}{t^2}$ annealing schedule on overall learning rate.
p_0, p_1, p_3	grad_on[k]	.5, .25, .25	probility of choosing a certain loss function
α_3	alpha_mask3	10% of cells	update only α_3 cells' tuning curves at a time

A.13 Optimization hyperparameters for Scheme B-2

Table A.6: Scheme B-2 – Simple scheme without denoiser

name	codename	value	description
U	num_freqs	200	number of frequencies
V	num_phis	2	number of phases
N_b	num_basis	UV	number of basis functions
N	num_cells	20	number of cells
P_{ij}^0	init_P[i,j]	$\mathcal{G}(0,1)$	coefficient matrix
$lr0$, $lr1$, $lr2$, $lr3$	lr0,lr1,lr2,lr3	.1 .1 .1 .01	baseline and relative learning rates for loss functions
$r_{z,tar}^2$	r2z1	0	target tuning negativity

Continued on next page

Table A.6 – continued from previous page

name	codename	value	description
w	w	1	intrinsic noise to set maximal frequency $1/w$
T	num_epochs	5e4	total training epochs
$\log L2_{tar}$	logl2tar	-4.6	
p_1, p_2, p_3	grad_on[k]	1/3, 1/3, 1/3	probility of choosing a certain loss function when $\log L2 \leqslant \log L2_{tar}$
p'_1, p'_2, p'_3	grad_on2[k]	1/5, 3/5, 1/5	probility of choosing a certain loss function when $\log L2 > \log L2_{tar}$
	rand01	0: 50% and 1: 50%	binary random number for evenly tuning on and off the gradient product terms $\nabla_1 \cdot \nabla_2$ and $\nabla_3 \cdot \nabla_2$ in SOGD.
α_3	alpha_mask3	20% of cells	update only α_3 cells' tuning curves at a time

A.14 Optimization hyperparameters for Scheme C

Table A.7: Scheme C – Simple scheme with new capacity measure

name	codename	value	description
L	num_periods	20	coding range
	num_points	10	number of discrete points in a period
U	num_freqs	200	number of frequencies
V	num_phis	2	number of phases
N_b	num_basis	UV	number of basis functions
N	num_cells	10~80	number of cells
P^0_{ij}	init_P[i,j]	$\mathcal{G}(0,1)$	coefficient matrix
$lr0$, $lr1$, $lr3$	lr0, lr1, lr3	.01, .01, 1	baseline and relative learning rates for loss functions
$r^2_{z,tar}$	r2z1	.001	target tuning negativity

Continued on next page

Table A.7 – continued from previous page

name	codename	value	description
w	`w`	.5~4	intrinsic noise to set maximal frequency $1/w$
T	`num_epochs`	1e5	total training epochs
β	`beta`	.1	last learning rate in annealing schedule
	`sch`	$\left[1 + \frac{1}{\beta - 1}\left(\frac{t - T/4}{T - T/4}\right)^2\right]^{-1}$	$\frac{1}{t^2}$ annealing schedule on overall learning rate.
p_0, p_1, p_3	`grad_on[k]`	.5, .25, .25	probility of choosing a certain loss function
α_3	`alpha_mask3`	10% of cells	update only α_3 cells' tuning curves at a time

Appendix B

Supplementary Information for Chapter 3

B.1 Interpolating codewords are always orthogonal to the distant landmark codewords

In the step of online similarity matching in the optimal online learning algorithm, interpolating codewords are generated between two adjacent orthogonal landmark codewords. To show how these interpolating codewords overlaps only to the each others and the two landmark codewords, I use a binary neuron example below as a simple proof of concept for which the sparsity is required to be conserved and the interpolating codewords are non-orthogonal to the two landmark codewords.

Consider p out of N cells being active for each codewords, without loss of generality, after re-indexing, two orthogonal landmark codewords can be

$$\mathbf{z}(x_1) = (1, \ldots, 1,\ 0, \ldots, 0,\ 0, \ldots, 0) \tag{B.1}$$

has Cell 1 to Cell p being active, and

$$\mathbf{z}(x_2) = (0, \ldots, 0,\ 1, \ldots, 1,\ 0, \ldots, 0) \tag{B.2}$$

has Cell $p+1$ to Cell $2p$ being active. Next, find a set of interpolating codewords:

$$\{\mathbf{z}(x) \mid x \in (x_1, x_2)\} \tag{B.3}$$

that incrementally connect $\mathbf{z}(x_1)$ and $\mathbf{z}(x_2)$. For which, in a binary neuron scheme, one can use a permutation matrix that preserves the number of active cells, move only one active cells out of the current active p cells, and create a locally translational invariant codeword segment:

$$\mathbf{z}(x+1) = P\ \mathbf{z}(x) \text{ for } x \in [x_1, x_2). \tag{B.4}$$

There are two criteria needed to be satisfied for z(x) to be an interpolating codewords. First, every interpolating codeword should be nearly parallel to *at least one* other codeword within the set: $\{\mathbf{z}(x) \mid x \in [x_1, x_2]\}$ such that

$$\mathbf{z}(x) \cdot \mathbf{z}(x') = p - 1. \tag{B.5}$$

Second, every interpolating codeword should *not* be orthogonal to any others within the set: $\{\mathbf{z}(x) \mid x \in [x_1, x_2]\}$ except $\mathbf{z}(x_1) \cdot \mathbf{z}(x_2) = 0$ such that

$$\mathbf{z}(x) \cdot \mathbf{z}(x')1. \tag{B.6}$$

To satisfy both, the permutation should take exactly p steps to go from one landmark codeword to the other such that

$$\mathbf{z}(x_2) = P^p \mathbf{z}(x_1). \tag{B.7}$$

Consequently, all codewords generated from the permutation, $\mathbf{z}(x_1{+}k) = P^k \mathbf{z}(x_1)$.where $0 < k < p$, must have p active cells only out of the first 2p active cells—which are the active cells of codewords $\mathbf{z}(x_1)$ and $\mathbf{z}(x_2)$. Below are an example with $\mathbf{z}(x_2) = P^{q=p}\mathbf{z}(x_1)$, and a counterexample with $\mathbf{z}(x_2) = P^{q=p+1}\mathbf{z}(x_1)$ for $p = 3$.

$$\begin{aligned} \mathbf{z}(x_1) &= (1,1,1,0,0,0,0,...,0) \\ \mathbf{z}(x_1+1) &= (0,1,1,1,0,0,0,...,0) \\ \mathbf{z}(x_1+2) &= (0,0,1,1,1,0,0,...,0) \\ \mathbf{z}(x_2) = \mathbf{z}(x_1+3) &= (0,0,0,1,1,1,0,...,0) \end{aligned} \tag{B.8}$$

From the above, $q = p$ with permuted cell index: $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to \cdots$,

so that the interpolating codewords satisfies $1 \leqslant \mathbf{z}(x) \cdot \mathbf{z}(x') \leqslant p - 1$.

$$\begin{aligned} \mathbf{z}(x_1) &= (1, 1, 1, 0, 0, 0, 0, ..., 0) \\ \mathbf{z}(x_1 + 1) &= (0, 1, 1, 0, 0, 0, 1, ..., 0) \\ \mathbf{z}(x_1 + 2) &= (0, 0, 1, 1, 0, 0, 1, ..., 0) \\ \mathbf{z}(x_1 + 3) &= (0, 0, 0, 1, 1, 0, 1, ..., 0) \\ \mathbf{z}(x_2) = \mathbf{z}(x_1 + 4) &= (0, 0, 0, 1, 1, 1, 0, ..., 0) \end{aligned} \tag{B.9}$$

From the above, $q = p + 1$ with permuted cell index: $1 \to 2 \to 3 \to 7 \to 4 \to 5 \to 6 \to \cdots$. Note that $\mathbf{z}(x_1) \cdot \mathbf{z}(x_1 + 3) = 0$ and $\mathbf{z}(x_2) \cdot \mathbf{z}(x_1 + 1) = 0$ which violate the non-orthogonality requirement $\mathbf{z}(x) \cdot \mathbf{z}(x')1$.

To conclude, a set of interpolating codewords from the process of *online similarity matching* only establish a link between two adjacent landmark codewords, and remain orthogonal to all the other landmark codewords.

B.2 Training hyperparameters for the sequential tasks

Table B.1: Sequential landmark-prediction tasks

name	codename	value	description
N_P	num_pc	256	
β	beta	$2/N_P$	
W_{PP}	Wpp	$\mathcal{G}(-2\beta, \beta)$ or $\mathbf{I} - \beta$	two types of initializations are used
W^0_{LP}	Wgg	0	
b_P	b_P	1	bias of place cell layer
b_L	b_L	1	bias of landmark prediction layer
a_L	cue_strength	.1	
a_v	vel_strength	.1	
U_v	Uv	linspace$(-.1, .1)$	vel. input weights
U_k	U_k	random.uniform$(-.5, .5)$	landmark input weights
η	eta	.1 ($\Delta t/\tau$ = 2ms/20ms)	Δt: discretized time-interval τ: membrane time constant
a_{rand}	a_rand	.000256	random acceleration strength
v_{max}	v_max	.0128 (.128cm/2ms = 64cm/s)	maximum speed
a	a	$a^{t+1} = \mathcal{G}(0, a_{rand})$	acceleration
x_v	x_k[1,t]	$v^{t+1} = \text{clip}\,(v^t + a^t, -v_{max}, v_{max})$	velocity inputs in k-th env.
r	x_k[0,t]	$r^{t+1} = (r^t + v^t) \mod L$	position on a ring in k-th env.
σ_L	lm_width	1	landmark input width
σ_{pred}	lmp_width	12	landmark-prediction signal width
$\mathrm{x}_k(t)$	x_k[2:,t]	$\left\{\exp\left(-\frac{\left(r(t)-r_l^{(k)}\right)^2}{\sigma_L^2}\right) \mid l = 1, \ldots, N_L^{(k)}\right\}$	landmark inputs in k-th env.
$\mathrm{x}_k^{\text{pred}}(t)$	y_k[:,t]	$\left\{\exp\left(-\frac{\left(r(t)-r_l^{(k)}\right)^2}{\sigma_{\text{pred}}^2}\right) \mid l = 1, \ldots, N_L^{(k)}\right\}$	landmark prediction signals in k-th env.
L	env_size	[64,101,84,57,115,43,89,87,64,112]$\times 10cm$	size of ten environments

Continued on next page

Table B.1 – continued from previous page

name	codename	value	description
r_l	lm_pos	[[0, 7, 22, 28, 48], [7, 14, 27, 39, 52, 71, 89, 99], [3, 12, 32, 45, 52, 60, 76], [9, 33, 43, 52], [4, 12, 28, 36, 45, 55, 62, 79, 100], [1, 10, 19], [13, 20, 40, 48, 56, 73, 86], [6, 17, 24, 39, 60, 74, 83], [6, 18, 38, 46, 57], [2, 10, 24, 33, 42, 54, 65, 79, 88]] $\times 10cm$	landmark positions of each environment
	ambi_lm_idx	[[ ], [ ]]; [[2, 5, 7], [ ]]; [[1, 5], [ ]]; [[ ], [ ]]; [[1, 3], [6, 8]]; [[ ], [ ]]; [[2, 5], [ ]]; [[ ], [ ]]; [[ ], [ ]]; [[2, 4, 7], [ ]]	index of indistinguishable landmarks in each environment
	batch_size	100	1 batch has 100 trials
	num_batches	10	1 epoch = 10 batches
T	timesteps	500 (500 * 2ms = 1s)	backprop timesteps
lr_{train}	learning_rate	.04	
lr_{adapt}	adapt_rate	.004	

B.3 Evaluating performance using the error of estimated position

Cross-validation simulation. In a training or testing session, the trainable weight matrices are stored every ten epochs for a subsequent cross-validation simulation, in which the weight are frozen and the agent random walks a different trajectory. The stored weight matrices are fed in every epoch such that the simulation takes 1/10 of the time for the original simulation.

Loss-to-error mapping. The landmark prediction error plotted in *cm* are calculated from the L2-norm loss function in a cross-validation simulation. The mapping from the loss function to the error is done by assuming that a displacement Δr from the ground truth r in Environment k is caused by a simple spatial shift of landmark-prediction signals, i.e. $\mathbf{s}_L^{(k)}(r(t)) \to \hat{\mathbf{x}}_k^{\text{pred}}(r(t), \Delta r)$, where

$$\hat{\mathbf{x}}_k^{\text{pred}}(r, \Delta r) \equiv \left\{ \exp\left(-\frac{\left(r - r_l^{(k)} - \Delta r\right)^2}{\sigma_{\text{pred}}^2} \right) \mid l = 1, \ldots, N_L^{(k)} \right\} \tag{B.10}$$

The corresponding loss function is

$$\hat{L}_k(\Delta r) \equiv \sum_r \frac{1}{2N_L^{(k)}} \left| \hat{\mathbf{x}}_k^{\text{pred}}(r, \Delta r) - \mathbf{x}_k^{\text{pred}}(r) \right|^2 \tag{B.11}$$

In the training scheme, I used a mapping averaging over ten environments:

$$\hat{L}(\Delta r) \equiv \frac{1}{10} \sum_k \hat{L}_k(\Delta r) \tag{B.12}$$

The error can thereby be numerically estimated from the loss from the training scheme.

$$\Delta r(t) \equiv \hat{L}^{-1}(L_k(t)) \tag{B.13}$$

The mapping function $\hat{L}^{-1}$ is plotted in Figure B.1.

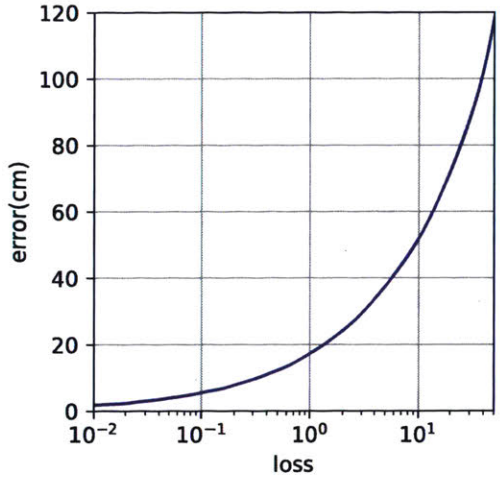

Figure B.1: Mapping between loss and error of estimated position.

B.4 LD vs HD manifolds in regular or topological environments

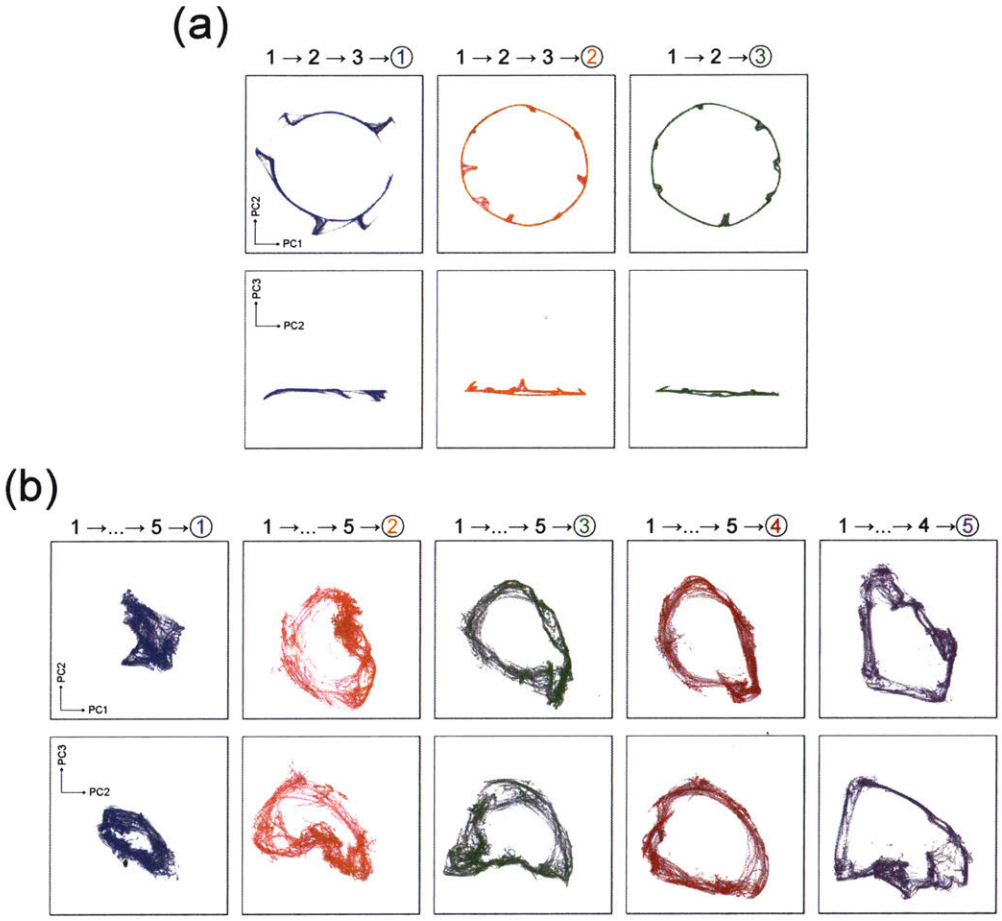

Figure B.2: Manifolds after learning multiple regular environments in the landmark-prediction task. **(a)** The manifolds of an LD network when navigating Environment 1,2, and 3 after successfully learning three environments in sequence. **(b)** The manifolds of an HD network when navigating Environment 1 to 5 after successfully learning five environments in sequence.

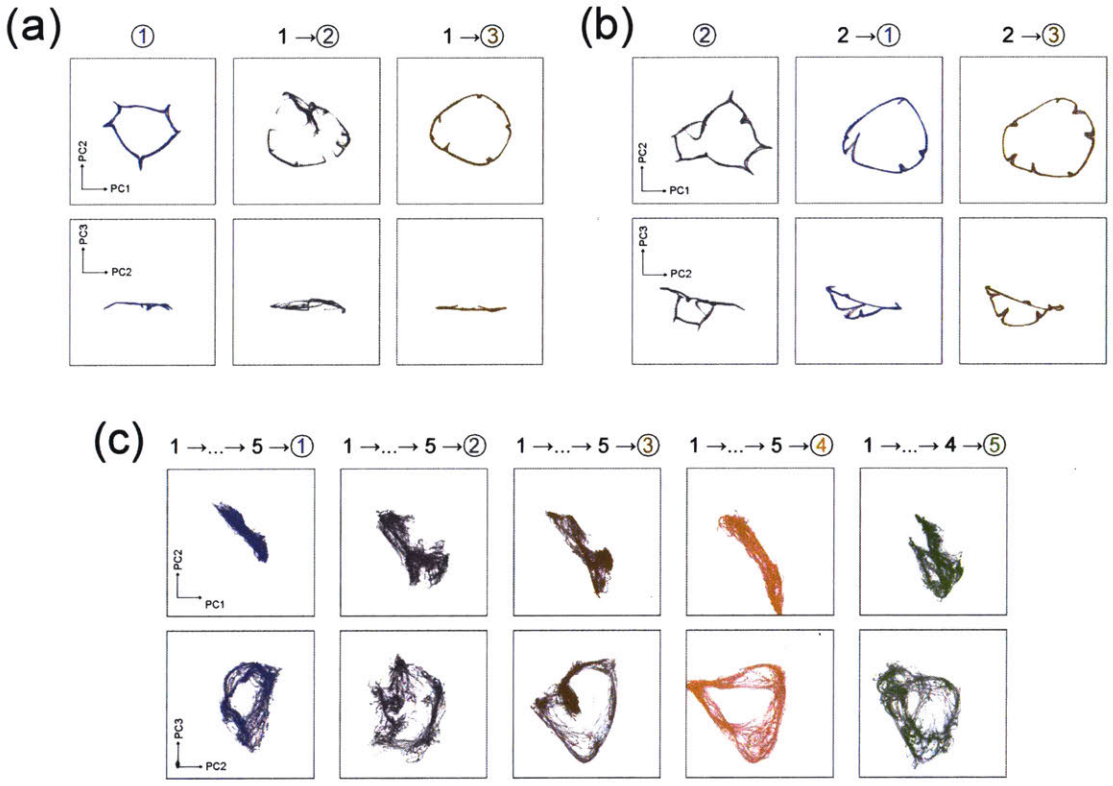

Figure B.3: Manifolds after learning multiple topological environments in the landmark-prediction task. **(a)** The manifolds of an LD network show a successful learning of Environment 1 and then Environment 3, but not Environment 2 (the manifold mistakenly connects the states representing multiple locations). **(b)** Although the learning of Environment 2 (a topologically complex environment) after learning Environment 1 (a topologically simple environment) failed to converge, learning Environment 2 directly from the original random initial connectivity succeeded. After learning a topologically complex environment, the subsequent learning of Environment 1 or 3 becomes easier. (c) The manifolds of an HD network when navigating Environment 1 to 5 after successfully learning five environments in sequence.

B.5 Learning ten environments

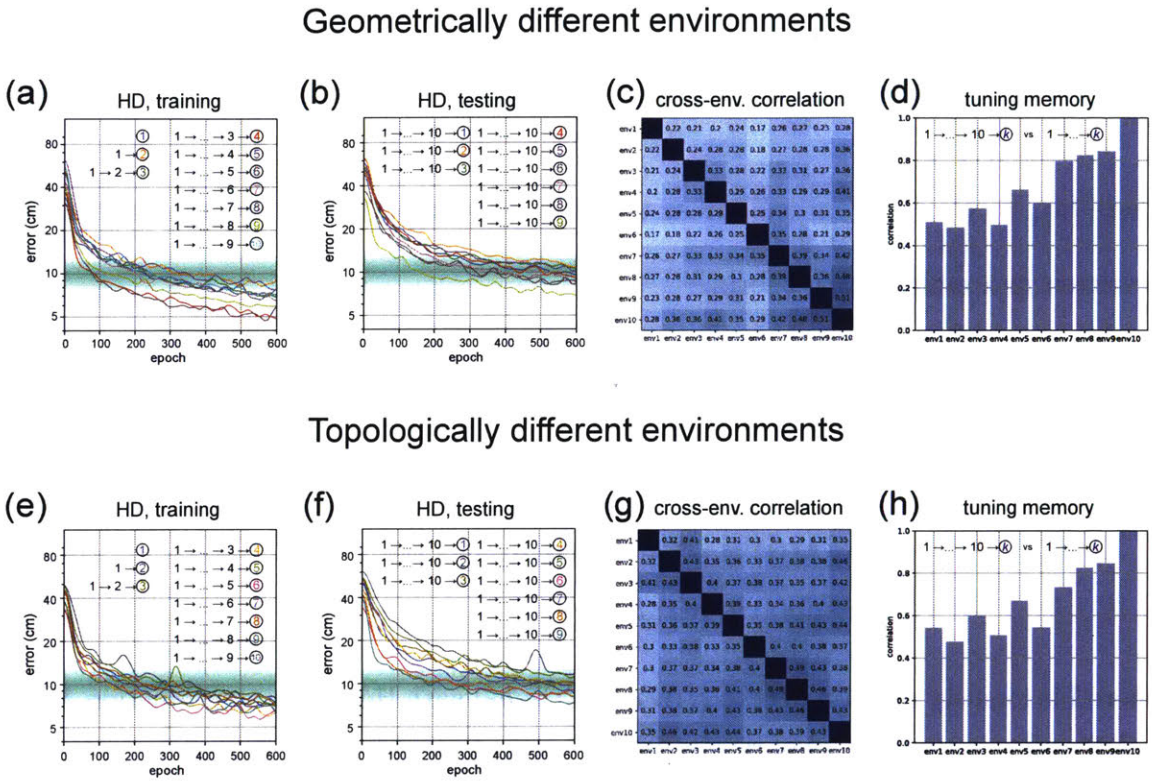

Figure B.4: An HD network can learn ten geometrically or topologically different environments. **(a-d)** From right to left are the learning curves, testing curves, cross-environment correlation matrix, and tuning memory for training in geometrically different environments. The tuning memory of Environment k measures the similarity between the tuning curves when they were first learnt and then later recalled after learning all ten environments. **(e-h)** Same as (a-d) but for the training in topologically different environments.

B.6 Computing relaxation time

The differential form of autonomous dynamical equation in the relaxation simulation is

$$\tau \frac{d\mathbf{s}_P(t)}{dt} = -\mathbf{s}_P(t) + [W_{PP}\mathbf{s}_P(t) + b_P]_+, \tag{B.14}$$

where $[u]_+ = \begin{cases} 0 & \text{if } u < 0 \\ 1 & \text{if } u \geqslant 0 \end{cases}$. Because the semi-linearity of above equation, it can be rewritten in a multiplicative form:

$$\tau \frac{d\mathbf{s}_P(t)}{dt} = M(t)\mathbf{s}_P(t) + \mathbf{b}(t). \tag{B.15}$$

The time dependent weight matrix and bias is defined as

$$M(t) \equiv -\mathbf{I} + [\mathbf{h}(t) \otimes \mathbf{h}(t)] \times W_{PP} \tag{B.16}$$

$$\mathbf{b}(t) \equiv \mathbf{h}(t)\ b_p(t), \tag{B.17}$$

where $\mathbf{I}$ is identity matrix, $\mathbf{h}(t) \equiv H[W_{PP}\mathbf{s}_P(t)+b_P]$ with Heaviside step function $H[\cdot]$, and $\otimes$ and $\times$ are operators of tensor product and element-wise multiplication, respectively. By applying eigen decomposition on $M(t) \equiv Q^{-1}(t)\Lambda(t)Q(t)$, Equation (B.15) becomes

$$\begin{aligned} \tau \frac{d}{dt} Q(t)\mathbf{s}_P(t) &= \Lambda(t)Q(t)\mathbf{s}_P(t) + Q(t)\mathbf{b}(t) \\ \Longrightarrow \tau \frac{d}{dt}\mathbf{a}(t) &= \Lambda(t)\mathbf{a}(t) + \mathbf{c}(t), \end{aligned} \tag{B.18}$$

where $\Lambda(t)$ is a diagonal matrix with eigenvalues: $\lambda_1(t) \leqslant \lambda_2(t) \leqslant \ldots \leqslant \lambda_{N_P}(t)$, $\mathbf{a}(t) \equiv Q(t)\mathbf{s}_P(t)$, and $\mathbf{c}(t) \equiv Q(t)\mathbf{b}(t)$. The instantaneous relaxation times can then be defined as the decay time of the transient terms of $\mathbf{a}(t)$, i.e. $a_i^{\text{tran}}(t) \propto e^{-\lambda_i t/\tau}$; the relaxation time for i-th mode at time t is

$$\tau_i^{\text{rel}}(t) \equiv \tau/\lambda_i \tag{B.19}$$

To get a sense of the stability of a continuous attractor, I plotted the k-th largest relaxation time. In case of a 1D continuous attractor, one can use $\tau_2^{\text{rel}}(t)$—for which $\tau_1^{\text{rel}}(t) \to \infty$ is relaxation time for the mode in longitudinal direction of a manifold. The smaller the $\tau_2^{\text{rel}}(t)$ the more stable the corresponding mode is.

B.7 Simulation hyperparameters for demonstrating persistent propensity

Table B.2: Demonstration of place cells' biased propensity

name	codename	value	description
T	num_steps	500*50 (50s)	total simulation time window
Δt	dt	2ms	discretize time-interval
Δ_{inter}	inter_input_duration	500 (1s)	
Δ_{pulse}	input_duration	50 (.1s)	
N_l	num_lm	T/Δ_{inter}	one input per second. Each input pulse last for .1s
x_l^t	x[t]	$[\mathcal{G}(0,1)]_+$	positive random inputs
N_P	num_cells	256	
N_{patt}	num_lm_stored	[0,30,100]	three levels of weight perturbation are simulated.
α_P	alpha	.9	
β_P	beta	$.3/N_P$	
ϵ	w_biased_str	3e-4	
W_{kWTA}	W_kWTA	$(\alpha_P + \beta_P)\,\mathbf{I} - \beta_P$	unbiased recurrent weight
$W(N_{patt})$	W(num_lm_stored)	$\sum_{\xi=1}^{N_{patts}} \mathbf{y}_\xi \mathbf{y}_\xi^T - \left\langle \mathbf{y}_\xi \mathbf{y}_\xi^T \right\rangle$, where $\mathbf{y}_\xi^{N_l \times 1} \sim \mathcal{G}(0,1)$	symmetric weight matrix that stores multiple patterns
W_{PP}	Wpp	$W_{kWTA} + \epsilon W(N_{patts})$	perturbed recurrent weight
b_P	b	$N_P\ \beta_P/2$	recurrent layer bias
η	eta	.1 ($\Delta t/\tau$ = 2ms / 20ms)	Δt: discretized time-interval; τ: membrane time constant

Bibliography

Alme, C. B., Miao, C., Jezek, K., Treves, A., Moser, E. I., and Moser, M.-B. (2014). Place cells in the hippocampus: Eleven maps for eleven rooms. *Proceedings of the National Academy of Sciences*, 111(52):18428–18435.

Babichev, A. and Dabaghian, Y. A. (2018). Topological Schemas of Memory Spaces. *Frontiers in Computational Neuroscience*, 12(April):1–16.

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., Pritzel, A., Chadwick, M. J., Degris, T., Modayil, J., Wayne, G., Soyer, H., Viola, F., Zhang, B., Goroshin, R., Rabinowitz, N., Pascanu, R., Beattie, C., Petersen, S., Sadik, A., Gaffney, S., King, H., Kavukcuoglu, K., Hassabis, D., Hadsell, R., and Kumaran, D. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433.

Baram, A. B., Muller, T. H., Whittington, J. C. R., and Behrens, T. E. (2018). Intuitive planning: global navigation through cognitive maps based on grid-like codes. *bioRxiv*, page 421461.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H., and Silver, D. (2016). Successor Features for Transfer in Reinforcement Learning. (Nips).

Barry, C., Hayman, R., Burgess, N., and Jeffery, K. J. (2007). Experience-dependent rescaling of entorhinal grids. *Nature Neuroscience*, 10(6):682–684.

Battaglia, F. P., Sutherland, G. R., and McNaughton, B. L. (2004). Local Sensory Cues and Place Cell Directionality: Additional Evidence of Prospective Coding in the Hippocampus. *Journal of Neuroscience*, 24(19):4541–4550.

Battaglia, F. P. and Treves, A. (1998). Attractor neural networks storing multiple space representations: A model for hippocampal place fields. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 58(6):7738–7753.

Behrens, T. E., Muller, T. H., Whittington, J. C., Mark, S., Baram, A. B., Stachenfeld, K. L., and Kurth-Nelson, Z. (2018). What Is a Cognitive Map? Organizing Knowledge for Flexible Behavior. *Neuron*, 100(2):490–509.

Bell, A. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–59.

Bell, A. J. and Sejnowski, T. J. (1997). The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338.

Bellec, G., Scherr, F., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2019). Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. pages 1–37.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTU:437–478.

Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., and Lin, Z. (2015). Towards Biologically Plausible Deep Learning. *arXiv preprint arXiv:1502.04156*.

Benna, M. K. and Fusi, S. (2016). Computational principles of synaptic memory consolidation. *Nature Neuroscience*, 19(12):1697–1706.

Bialek, W. (2012). Photon Counting in Vision. In *Biophysics: searching for principles*. Princeton University Press.

Bittner, K. C., Milstein, A. D., Grienberger, C., Romani, S., and Magee, J. C. (2017). Behavioral time scale synaptic plasticity underlies CA1 place fields. *Science*, 357(6355):1033–1036.

Bonnevie, T., Dunn, B., Fyhn, M., Hafting, T., Derdikman, D., Kubie, J. L., Roudi, Y., Moser, E. I., and Moser, M. B. (2013). Grid cells require excitatory drive from the hippocampus. *Nature Neuroscience*, 16(3):309–317.

Bottou, L. (2012). Stochastic gradient descent tricks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTU(1):421–436.

Brandon, M. P., Koenig, J., Leutgeb, J. K., and Leutgeb, S. (2014). New and Distinct Hippocampal Place Codes Are Generated in a New Environment during Septal Inactivation. *Neuron*, 82(4):789–796.

Brea, J., Gaál, A. T., Urbanczik, R., and Senn, W. (2016). Prospective Coding by Spiking Neurons. *PLoS Computational Biology*, 12(6):1–25.

Brun, V. H., Solstad, T., Kjelstrup, K. B., Fyhn, M., Witter, M. P., Moser, E. I., and Moser, M. B. (2008). Progressive increase in grid scale from dorsal to ventral medial entorhinal cortex. *Hippocampus*, 18(12):1200–1212.

Burak, Y. and Fiete, I. R. (2009). Accurate path integration in continuous attractor network models of grid cells. *PLoS Computational Biology*, 5(2).

Burak, Y. and Fiete, I. R. (2012). Fundamental limits on persistent activity in networks of noisy neurons. *Proc. Natl. Acad. Sci.*, 109(43):17645–17650.

Bush, D., Barry, C., Manson, D., and Burgess, N. (2015). Using Grid Cells for Navigation. *Neuron*, 87(3):507–520.

Chen, Y., Paiton, D. M., and Olshausen, B. A. (2018). The Sparse Manifold Transform. *arXiv*, page 1806.08887.

Chen, Z., Gomperts, S. N., Yamamoto, J., and Wilson, M. A. (2014). Neural representation of spatial topology in the rodent hippocampus. *Neural Computation*, 26(1):1–39.

Chen, Z., Kloosterman, F., Brown, E. N., and Wilson, M. A. (2012). Uncovering spatial topology represented by rat hippocampal population neuronal codes. *Journal of Computational Neuroscience*, 33(2):227–255.

Chung, S., Lee, D. D., and Sompolinsky, H. (2018). Classification and Geometry of General Perceptual Manifolds. *Physical Review X*, 8(3):31003.

Cohen, U., Chung, S., Lee, D. D., and Sompolinsky, H. (2019). Separability and Geometry of Object Manifolds in Deep Neural Networks. *bioRxiv*, page 644658.

Colgin, L. L. (2016). Rhythms of the hippocampal network. *Nature Reviews Neuroscience*, 17:239.

Colgin, L. L., Moser, E. I., and Moser, M. B. (2008). Understanding memory through hippocampal remapping. *Trends in Neurosciences*, 31(9):469–477.

Coop, R. and Arel, I. (2013). Mitigation of catastrophic forgetting in recurrent neural networks using a Fixed Expansion Layer. *Proceedings of the International Joint Conference on Neural Networks*.

Cox, R. T. and Carlton, C. E. (2003). A Comment on Gene Introgression versus En Masse Cycle Switching in the Evolution of 13-Year and 17-Year Life Cycles in Periodical Cicadas. *Evolution*, 57(2):428–432.

Cueva, C. J. and Wei, X.-X. (2018). Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *ICLR*, pages 1–19.

Curto, C. (2016). What can topology tell us about the neural code? *Bulletin of the American Mathematical Society*, 54(1):63–78.

Curto, C., Gross, E., Jeffries, J., Morrison, K., Omar, M., Rosen, Z., Shiu, A., and Youngs, N. (2017). What makes a neural code convex? *SIAM J. Appl. Algebr. Geom.*, 1(1):222–238.

Curto, C. and Itskov, V. (2008). Cell groups reveal structure of stimulus space. *PLoS Computational Biology*, 4(10).

Dabaghian, Y., Brandt, V. L., and Frank, L. M. (2014). Reconceiving the hippocampal map as a topological template. *eLife*, 3:e03476.

Dabaghian, Y., Mémoli, F., Frank, L., and Carlsson, G. (2012). A Topological Paradigm for Hippocampal Spatial Map Formation Using Persistent Homology. *PLoS Computational Biology*, 8(8).

Daugman, J. G. (1988). Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1169–1179.

Davoudi, H. and Foster, D. J. (2019). Acute silencing of hippocampal CA3 reveals a dominant role in place field responses. *Nature Neuroscience*, 22(3):337–342.

Dayan, P. (1993). Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 5(4):613–624.

DiCarlo, J. J. and Cox, D. D. (2007). Untangling invariant object recognition. *Trends in Cognitive Sciences*, 11(8):333–341.

DiCarlo, J. J., Zoccolan, D., and Rust, N. C. (2012). How does the brain solve visual object recognition? *Neuron*, 73(3):415–434.

Dordek, Y., Soudry, D., Meir, R., and Derdikman, D. (2016). Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis. *eLife*, 5(MARCH2016):1–36.

Druckmann, S. and Chklovskii, D. B. (2012). Neuronal circuits underlying persistent representations despite time varying activity. *Current Biology*, 22(22):2095–2103.

Dupret, D., O'Neill, J., Pleydell-Bouverie, B., and Csicsvari, J. (2010). The reorganization and reactivation of hippocampal maps predict spatial memory performance. *Nat. Neurosci.*, 13(8):995–1002.

Erdem, U. M. and Hasselmo, M. (2012). A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *European Journal of Neuroscience*, 35(6):916–931.

Erdem, U. M. and Hasselmo, M. E. (2014). A biologically inspired hierarchical goal directed navigation model. *Journal of Physiology Paris*, 108(1):28–37.

Ermentrout, G. B. and Terman, D. H. (2010). Firing Rate Models ermentrout. In *Mathematical foundations of neuroscience*, volume 35, chapter 11. Springer Science & Business Media.

Fiete, I. R., Burak, Y., and Brookings, T. (2008). What Grid Cells Convey about Rat Location. *Journal of Neuroscience*, 28(27):6858–6871.

Frank, L. M., Stanley, G. B., and Brown, E. N. (2004). Hippocampal Plasticity across Multiple Days of Exposure to Novel Environments. *Journal of Neuroscience*, 24(35):7681–7689.

Fuhs, M. C. (2006). A Spin Glass Model of Path Integration in Rat Medial Entorhinal Cortex. *Journal of Neuroscience*, 26(16):4266–4276.

Fyhn, M., Hafting, T., Treves, A., Moser, M. B., and Moser, E. I. (2007). Hippocampal remapping and grid realignment in entorhinal cortex. *Nature*, 446(7132):190–194.

Gardner, M. P. H., Schoenbaum, G., and Gershman, S. J. (2018). Rethinking dopamine as generalized prediction error. *bioRxiv*.

Gardner, R. J., Lu, L., Wernle, T., Moser, M.-B., and Moser, E. I. (2019). Correlation structure of grid cells is preserved during sleep. *Nature Neuroscience*, 22(4):598–608.

Gershman, S. J. (2018). The Successor Representation: Its Computational Logic and Neural Substrates. *The Journal of Neuroscience*, 38(33):7193–7200.

Gil, M., Ancau, M., Schlesiger, M. I., Neitz, A., Allen, K., De Marco, R. J., and Monyer, H. (2018). Impaired path integration in mice with disrupted grid cell firing. *Nature Neuroscience*, 21(1):81–93.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Grossberg, B. S. (1973). Contour Enhancement , Short Term Memory ,. *Studies in Applied Mathematics*, L11(September).

Grossberg, S. (1988). Nonlinear Neural Networks: Principles, Mechanisms, and Architectures. *Neural Networks*, I:17–61.

Gu, Y., Lewallen, S., Kinkhabwala, A. A., Domnisoru, C., Yoon, K., Gauthier, J. L., Fiete, I. R., and Tank, D. W. (2018). A Map-like Micro-Organization of Grid Cells in the Medial Entorhinal Cortex. *Cell*, 175(3):736–750.

Guzman, S. J., Schlögl, A., Frotscher, M., and Jonas, P. (2016). Synaptic mechanisms of pattern completion in the hippocampal CA3 network. *Science*, 353(6304):1117–1123.

Hafting, T., Fyhn, M., Molden, S., Moser, M. B., and Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806.

Hedrick, K. R. and Zhang, K. (2016). Megamap: flexible representation of a large space embedded with nonspatial information by a hippocampal attractor network. *Journal of Neurophysiology*, 116(2):868–891.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Herz, A. V., Mathis, A., and Stemmler, M. (2017). Periodic population codes: From a single circular variable to higher dimensions, multiple nested scales, and conceptual spaces. *Current Opinion in Neurobiology*, 46:99–108.

Hollup, S. A., Molden, S., Donnett, J. G., Moser, M. B., and Moser, E. I. (2001). Accumulation of hippocampal place fields at the goal location in an annular watermaze task. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 21(5):1635–44.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Biophysics*, 81(May):3088–3092.

Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215–243.

Kanitscheider, I. and Fiete, I. R. (2016). Training recurrent networks to generate hypotheses about how the brain solves hard navigation problems. *arXiv*.

Kanitscheider, I. and Fiete, I. R. (2017). Making our way through the world: Towards a functional understanding of the brain's spatial circuits. *Curr. Opin. Syst. Biol.*, 3:186–194.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2016). Overcoming catastrophic forgetting in neural networks. *arXiv*, page 1612.00796v2.

Kjelstrup, K. B., Solstad, T., Brun, V. H., Hafting, T., Leutgeb, S., Witter, M. P., Moser, E. I., and Moser, M.-B. (2008). Finite Scale of Spatial Representation in the Hippocampus. *Science*, 321(5885):140–143.

Krakauer, J. W., Ghazanfar, A. A., Gomez-Marin, A., MacIver, M. A., and Poeppel, D. (2017). Neuroscience Needs Behavior: Correcting a Reductionist Bias. *Neuron*, 93(3):480–490.

Kriener, B., Chaudhuri, R., and Fiete, I. R. (2017). How fast is neural winner-take-all when deciding between many options ? *bioRxiv*, pages 1–30.

Kropff, E. and Treves, A. (2008). The emergence of grid cells: Intelligent design or just adaptation? *Hippocampus*, 18(12):1256–1269.

Kubie, J. L. and Fenton, A. A. (2012). Linear look-ahead in conjunctive cells: an entorhinal mechanism for vector-based navigation. *Frontiers in neural circuits*, 6:20.

Kudrimoti, H. S., Barnes, C. A., and McNaughton, B. L. (1999). Reactivation of Hippocampal Cell Assemblies: Effects of Behavioral State, Experience, and EEG Dynamics. *The Journal of Neuroscience*, 19(10):4090–4101.

Kulkarni, T. D., Saeedi, A., Gautam, S., and Gershman, S. J. (2016). Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

Langston, R. F., Ainge, J. A., Couey, J. J., Canto, C. B., Bjerknes, T. L., Witter, M. P., Moser, E. I., and Moser, M.-B. (2010). Development of the Spatial Representation System in the Rat. *Science*, 328(5985):1576–1580.

Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv*, abs/1504.0.

Lee, J. S., Briguglio, J., Romani, S., and Lee, A. K. (2019). The statistical structure of the hippocampal code for space as a function of time, context, and value. *bioRxiv*, page 615203.

Leutgeb, J. K., Leutgeb, S., Moser, M.-B., and Moser, E. I. (2007). Pattern Separation in the Dentate Gyrus and CA3 of the Hippocampus. *Science*, 315(5814):961–966.

Leutgeb, J. K., Mankin, E. A., and Leutgeb, S. (2013). Population Coding by Place Cells and Grid Cells. In *Principles of Neural Coding*, pages 300–317. CRC Press.

Leutgeb, S. and Leutgeb, J. K. (2007). Pattern separation, pattern completion, and new neuronal codes within a continuous CA3 map. *Learning and Memory*, 14(11):745–757.

Leutgeb, S., Leutgeb, J. K., Moser, E. I., and Moser, M.-B. (2006). Fast rate coding in hippocampal CA3 cell ensembles. *Hippocampus*, 16(9):765–774.

Leutgeb, S., Leutgeb, J. K., Treves, A., Moser, M.-B. B., and Moser, E. I. (2004). Distinct Ensemble Codes in Hippocampal Areas CA3 and CA1. *Science*, 305(5688):1295–1298.

Low, R. J., Lewallen, S., Aronov, D., Nevers, R., and Tank, D. W. (2018). Probing variability in a cognitive map using manifold inference from neural dynamics. *bioRxiv*.

MacDonald, C. J., Lepage, K. Q., Eden, U. T., and Eichenbaum, H. (2011). Hippocampal "time cells" bridge the gap in memory for discontiguous events. *Neuron*, 71(4):737–749.

Majani, E., Erlarson, R., and Abu-Mostafa, Y. (1989). On the k-winners-takes-all network. *Advanced in Neural Information Processing Systems I*, pages 634–642.

Marr, D. (1971). Simple Memory: A Theory for Archicortex. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 262(841):23–81.

Marr, D. and Poggio, T. (1976). From understanding computation to understanding neural circuitry. *Neurosciences Research Program Bulletin*, 15(3):470–488.

Mathis, A., Herz, A. V., and Stemmler, M. B. (2013). Multiscale codes in the nervous system: The problem of noise correlations and the ambiguity of periodic scales. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 88(2):1–10.

Maurer, A. P., VanRhoads, S. R., Sutherland, G. R., Lipa, P., and McNaughton, B. L. (2005). Self-motion and the origin of differential spatial scaling along the septo-temporal axis of the hippocampus. *Hippocampus*, 15(7):841–852.

McHugh, T. J., Jones, M. W., Quinn, J. J., Balthasar, N., Coppari, R., Elmquist, J. K., Lowell, B. B., Fanselow, M. S., Wilson, M. A., and Tonegawa, S. (2007). Dentate Gyrus NMDA Receptors Mediate Rapid Pattern Separation in the Hippocampal Network. *Science*, 317(July):94–99.

McNaughton, B. L., Barnes, C. A., Gerrard, J. L., Gothard, K., Jung, M. W., Knierim, J. J., Kudrimoti, H., Qin, Y., Skaggs, W. E., Suster, M., and Weaver, K. L. (1996). Deciphering the hippocampal polyglot: the hippocampus as a path integration system. *Journal of Experimental Biology*, 199(1):173–185.

McNaughton, B. L., Barnes, C. A., and O'Keefe, J. (1983). The contributions of position, direction, and velocity to single unit activity in the hippocampus of freely-moving rats. *Experimental Brain Research*, 52(1):41–49.

Mehta, M. R., Quirk, M. C., and Wilson, M. A. (2000). Experience-dependent asymmetric shape of hippocampal receptive fields. *Neuron*, 25(3):707–715.

Mitchison, G. J. (1977). Phyllotaxis and the Fibonacci Series. *Science*, 196(4287):270–275.

Morris, R. G. M., Garrud, P., Rawlins, J. N. P., and O'Keefe, J. (1982). Place navigation impaired in rats with hippocampal lesions. *Nature*, 297(5868):681–683.

Moser, M. B., Rowland, D. C., and Moser, E. I. (2015). Place cells, grid cells, and memory. *Cold Spring Harbor Perspectives in Biology*, 7(2):a021808.

Muller, R. U., Kubie, J. L., and Ranck, J. B. (1987). Spatial firing patterns of hippocampal complex-spike cells in a fixed environment. *The Journal of Neuroscience*, 7(7):1935–1950.

Nakazawa, K., Quirk, M. C., Chitwood, R. A., Watanabe, M., Yeckel, M. F., Sun, L. D., Kato, A., Carr, C. A., Johnston, D., Wilson, M. A., and Tonegawa, S. (2002). Requirement for Hippocampal CA3 NMDA Receptors in Associative Memory Recall. *Science*, 297(5579):211–218.

Norimoto, H., Makino, K., Gao, M., Shikano, Y., Okamoto, K., Ishikawa, T., Sasaki, T., Hioki, H., Fujisawa, S., and Ikegaya, Y. (2018). Hippocampal ripples down-regulate synapses. *Science*, 359(6383):1524–1527.

O'Keefe, J., Burgess, N., Donnett, J. G., Jeffery, K. J., and Maguire, E. A. (1998). Place cells, navigational accuracy, and the human hippocampus. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 353(1373):1333–1340.

O'Keefe, J. and Nadel, L. (1978). *The hippocampus as a cognitive map*. Oxford: Clarendon Press.

Pehlevan, C., Hu, T., and Chklovskii, D. B. (2015). A Hebbian/Anti-Hebbian Neural Network for Linear Subspace Learning: A Derivation from Multidimensional Scaling of Streaming Data. *Neural Computation*, 27(7):1461–1495.

Pehlevan, C., Sengupta, A. M., and Chklovskii, D. B. (2018). Why Do Similarity Matching Objectives Lead to Hebbian/Anti-Hebbian Networks? *Neural Computation*, 30(1):84–124.

Poggio, T. (2012). The levels of understanding framework, revised. *Perception*, 41(9):1017–1023.

Pouget, A., Deneve, S., Ducom, J. C., and Latham, P. E. (1999). Narrow versus wide tuning curves: What's best for a population code? *Neural Computation*, 11(1):85–90.

Redish, A. D., Battaglia, F. P., Chawla, M. K., Ekstrom, A. D., Gerrard, J. L., Lipa, P., Rosenzweig, E. S., Worley, P. F., Guzowski, J. F., McNaughton, B. L., and Barnes, C. A. (2001). Independence of Firing Correlates of Anatomically Proximate Hippocampal Pyramidal Cells. *Journal of Neuroscience*, 21(5):RC134–RC134.

Rich, P. D., Liaw, H. P., and Lee, A. K. (2014). Large environments reveal the statistical structure governing hippocampal representations. *Science*, 345(6198):814–817.

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025.

Rolls, E. T. (2013). The mechanisms for pattern completion and pattern separation in the hippocampus. *Frontiers in Systems Neuroscience*, 7(October):1–21.

Rolls, E. T., Stringer, S. M., and Elliot, T. (2006). Entorhinal cortex grid cells can map to hippocampal place cells by competitive learning. *Network: Computation in Neural Systems*, 17(4):447–465.

Rumelhart, D. E. and Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive science*, 9(1):75–112.

Samsonovich, A. and McNaughton, B. L. (1997). Path Integration and Cognitive Mapping in a Continuous Attractor Neural Network Model. *Journal of Neuroscience*, 17(15):5900–5920.

Scellier, B. and Bengio, Y. (2016). Equilibrium Propagation: Bridging the Gap Between Energy-Based Models and Backpropagation. 11(May).

Schiller, J., Berlin, S., and Derdikman, D. (2018). The Many Worlds of Plasticity Rules. *Trends in Neurosciences*, 41(3):124–127.

Sengupta, A. M., Tepper, M., Pehlevan, C., Genkin, A., and Chklovskii, D. B. (2018). Manifold-tiling Localized Receptive Fields are Optimal in Similarity-preserving Neural Networks. *bioRxiv*.

Shoval, O., Sheftel, H., Shinar, G., Hart, Y., Ramote, O., Mayo, A., Dekel, E., Kavanagh, K., and Alon, U. (2012). Evolutionary trade-offs, pareto optimality, and the geometry of phenotype space. *Science*, 336(6085):1157–1160.

Solstad, T., Moser, E. I., and Einevoll, G. T. (2006). From Grid Cells to Place Cells: A Mathematical Model. *Hippocampus*, 16:1026–1031.

Sompolinsky, H., Crisanti, A., and Sommers, H. J. (1988). Chaos in random neural networks. *Phys. Rev. Lett.*, 61(3):259–262.

Sreenivasan, S. and Fiete, I. (2011). Grid cells generate an analog error-correcting code for singularly precise neural computation. *Nature Neuroscience*, 14(10):1330–1337.

Stachenfeld, K. L., Botvinick, M. M., and Gershman, S. J. (2017). The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653.

Stemmler, M., Mathis, A., and Herz, A. V. (2015). Connecting multiple spatial scales to decode the population activity of grid cells. *Science Advances*, 1(11):1–12.

Stensola, H., Stensola, T., Solstad, T., FrØland, K., Moser, M. B., and Moser, E. I. (2012). The entorhinal grid map is discretized. *Nature*, 492(7427):72–78.

Stensola, T., Stensola, H., Moser, M. B., and Moser, E. I. (2015). Shearing-induced asymmetry in entorhinal grid cells. *Nature*, 518(7538):207–212.

Tai Sing Lee (1996). Image representation using 2D Gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):959–971.

Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. pages 1–16.

Trettel, S. (2017). Grid cell co-activity patterns remain stable across different behavioral states and experiences.

Trettel, S. G., Trimper, J. B., Hwaun, E., Fiete, I. R., and Colgin, L. L. (2019). Grid cell co-activity patterns during sleep reflect spatial overlap of grid fields during active behaviors. *Nature Neuroscience*, 22(4):609–617.

Treves, A., Rolls, E. T., and Simmen, M. (1997). Time for retrieval in recurrent associative memories. *Physica D: Nonlinear Phenomena*, 107(2):392–400.

Tsodyks, M. (2005). Attractor neural networks and spatial maps in hippocampus. *Neuron*, 48(2):168–169.

van Vreeswijk, C. (2001). Whence Sparseness? In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 180–186. MIT Press.

Whittington, J. C. R., Muller, T. H., Mark, S., Barry, C., and Behrens, T. E. J. (2018). Generalisation of structural knowledge in the hippocampal-entorhinal system. (Nips).

Widloski, J. and Fiete, I. R. (2014). A model of grid cell development through spatial exploration and spike time-dependent plasticity. *Neuron*, 83(2):481–495.

Wills, T. J., Cacucci, F., Burgess, N., and O'Keefe, J. (2010). Development of the hippocampal cognitive map in preweanling rats. *Science (New York, N.Y.)*, 328(5985):1573–6.

Wills, T. J., Lever, C., Cacucci, F., Burgess, N., and O'Keefe, J. (2005). Attractor dynamics in the hippocampal representation of the local environment. *Science*, 308(5723):873–876.

Wilson, F. (1991). Purpose and Function in Biology. In *Empiricism and Darwin's Science*, pages 131–162. Springer Netherlands, Dordrecht.

Wilson, M. A. and McNaughton, B. L. (1993). Dynamics of the hippocampal ensemble code for space. *Science*, 261:1055–1058.

Wilson, M. A. and McNaughton, B. L. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172):676–679.

Witter, M. P. and Amaral, D. G. (2004). Hippocampal Formation. In *The Rat Nervous System*, pages 635–704. Elsevier Inc.

Xie, X., Hahnloser, R. H., and Sebastian Seung, H. (2002). Selectively grouping neurons in recurrent networks of lateral inhibition. *Neural Computation*, 14(11):2627–2646.

Yoo, Y. S. (2014). *Multi-scale error-correcting codes and their decoding using belief propagation*. PhD thesis.

Yoon, K., Buice, M. A., Barry, C., Hayman, R., Burgess, N., and Fiete, I. R. (2013). Specific evidence of low-dimensional continuous attractor dynamics in grid cells. *Nature Neuroscience*, 16(8):1077–1084.

Zhang, K., Ginzburg, I., McNaughton, B. L., and Sejnowski, T. J. (1998). Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells. *Journal of neurophysiology*, 79(2):1017–44.

Zhang, K. and Sejnowski, T. J. (1999). Neuronal tuning: To sharpen or broaden? *Neural Computation*, 11(1):75–84.

Zheng, C. and Colgin, L. L. (2018). Hippocampal slow and fast gamma rhythms correlate differentially with successful memory performance in a goal-directed spatial memory task. In *SfN*.

Ziv, Y., Burns, L. D., Cocker, E. D., Hamel, E. O., Ghosh, K. K., Kitch, L. J., Gamal, A. E., and Schnitzer, M. J. (2013). Long-term dynamics of CA1 hippocampal place codes. *Nat. Neurosci.*, 16(3):264–266.